VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Simulace provozu technologie LoRa pomocí nástroje FLoRa a OMNeT++

LoRa Technology Simulation Using FLoRa and OMNeT++

Vladimír Dobeš

Bakalářská práce

Vedoucí práce: Ing. Libor Michalek, Ph.D.

Ostrava, 2021

Abstrakt

Cílem práce je popsat a zdokumentovat práci se simulační knihovnou FloRa v prostředí OMNeT++ a na základě získaných poznatků vytvořit dvě laboratorní úlohy do odborného předmětu Modelování sítí. V úvodu je čtenář seznámen s Low-Power Wide Area sítěmi a s technologiemi LoRa a LoRaWAN. Dále se práce věnuje popisu a zprovoznění prostředí OMNeT++ společně s jeho rozšířením INET framework. Poslední část obsahuje dvě vypracované laboratorní úlohy, které pozorují vlivu tzv. Adaptive Data Rate mechanismu na chod LoRaWAN sítě.

Klíčová slova

LPWAN, LoRa, LoRaWAN, OMNeT++, INET, FloRa

Abstract

The thesis aims to describe and document the work possibilities of the simulation library FloRa in the OMNeT++ environment. With acquired information, two laboratory assignments into the Network Modeling course are created. In the introduction, Low-Power Wide-Area networks and LoRa and LoRaWAN technologies are described. Furthermore, the work deals with the description and commissioning of the OMNeT++ environment together with its extension INET framework. The last part contains two laboratory tasks that observe the influence of the so-called Adaptive Data Rate mechanism on the LoRaWAN network.

Keywords

LPWAN, LoRa, LoRaWAN, OMNeT++, INET, FloRa

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Liborovi Michalkovi, Phd. za odbornou pomoc, vstřícnost při konzultacích a rady při zpracování této práce.

Obsah

Seznam použitých symbolů a zkratek **6**

Seznam obrázků **8**

Seznam tabulek **10**

1 Úvod **11**

2 Low-Power Wide Area Networks **12**

3 LoRa **13**

3.1 Modulace 13

3.2 Spreading Factor 14

4 LoRaWAN **15**

4.1 Síťová architektura 16

4.2 Třídy 16

4.3 Regulace 17

4.4 Formát zprávy 18

4.5 Adaptive Data Rate 18

4.6 Řešení kolizí 20

4.7 Zabezpečení 22

5 OMNeT++ a INET framework **23**

5.1 Diskrétní události 23

5.2 Moduly 24

5.3 Stavba OMNeT++ 24

5.4 OMNeT++ IDE 25

5.5 INET Framework 28

5.6 Instalace OMNeT++ 28

6 FloRa Framework **31**
6.1 Základní informace 31
6.2 Spotřeba elektrické energie 32
6.3 Síťová architektura 32
6.4 ADR+ 32
6.5 Předem předpřipravené simulace 33
6.6 Přidání FloRa framework do OMNeT++ 34

7 Zprovoznění simulací **35**
7.1 LoRaWAN síť s vypnutým ADR mechanismem 35
7.2 LoRaWAN síť se zapnutým ADR mechanismem 42

8 Závěr **46**

Literatura **47**

Přílohy **48**

A Konfigurace backhaul sítě **49**

B Konfigurace spotřeby energie **50**

C Konfigurace v souboru package.ned **51**

D Konfigurace v souboru omnetpp.ini **53**

Seznam použitých zkratek a symbolů

IoT	–	Internet of Things
LPWAN	–	Low-Power Wide Area Network
LPWA	–	Low-Power Wide Area
MAC	–	Media Access Protocol
TDMA	–	Time Division Multiple Access
CSS	–	Chirp Spread Spectrum
FSK	–	Frequency Shift Keying
SF	–	Spreading Factor
ToA	–	Time on Air
QoS	–	Quality of Service
IP	–	Internet Protocol
RSSI	–	Received Signal Strength Indicator
AES	–	Advanced Encryption Standard
ETSI	–	European Telecommunications Standards Institute
LBT AFA	–	Listen Before Talk Adaptive Frequency Agillity
ADR	–	Adaptive Data Rate
ABP	–	Activation By Personalization
OTAA	–	Over-the-air activation
NED	–	Network Description
IDE	–	Integrated Development Environment
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
IPv4	–	Internet Protocol version 4
IPv6	–	Internet Protocol version 6
OSPF	–	Open Shortest Path First
BGP	–	Border Gateway Protocol
PPP	–	Point-to-Point Protocol
MANET	–	Mobile Ad Hoc Network

MPLS	–	Multi-Protocol Label Switching
LDP	–	Label Distribution Protocol
RVSP-TE	–	Resource Reservation Protocol - Traffic Engineering
LTE	–	Long Term Evolution
LTS	–	Long Term Support
SNR	–	Signal-to-Noise Ratio
DER	–	Data Extraction Rate

Seznam obrázků

3.1 Up-chirp (vlevo) a down-chirp (vpravo) pulzy 13

4.1 Duty-cycle 20% 17
4.2 Formát zprávy 18
4.3 ADR mechanismus 19
4.4 Pole frame control 19
4.5 Neúspěšný ADR mechanismus 20
4.6 Mechanismus pro potvrzování zprávy 21

5.1 Stavba modulů 24
5.2 OMNeT++ IDE 26
5.3 Výsledkové soubory a jejich převod do .anf souboru 27
5.4 Výsledný .anf soubor s filtrem 27
5.5 Nabídka pro instalaci INET Framework 30

6.1 loRaNetworkTest.ini 34

7.1 Náhled konstruované LoRaWAN sítě 36
7.2 Spuštění simulace 37
7.3 Nabídka Select Executable 37
7.4 Moduly energyConsumer a SimpleLoRaApp 38
7.5 LoRaWAN síť s vypnutým ADR mechanismem - počet odeslaných paketů na zařízení 39
7.6 LoRaWAN síť s vypnutým ADR mechanismem - spotřeba energie pro jednotlivé rozprostírací faktory 39
7.7 Modul radio 40
7.8 Modul udpApp 40
7.9 LoRaWAN síť s vypnutým ADR mechanismem - RSSI histogram 41
7.10 LoRaWAN síť se zapnutým ADR mechanismem - počet odeslaných paketů na zařízení 43
7.11 LoRaWAN síť se zapnutým ADR mechanismem - spotřeba energie pro jednotlivé rozprostírací faktory 43

7.12 LoRaWAN síť se zapnutým ADR mechanismem - počet přijatých ADR paketů na zařízení 44
7.13 LoRaWAN síť se zapnutým ADR mechanismem - RSSI histogram 45

Seznam tabulek

3.1 Citlivost pro rozprostírací faktory při šířce pásma 125 KHz 14

4.1 EU863-870 DataRate 15

6.1 FloRa parametry pro prostředí 32

7.1 LoRaWAN síť s vypnutým ADR mechanismem - rozložení rozprostíracího faktoru . . 38
7.2 LoRaWAN síť s vypnutým ADR mechanismem - DER a počet přijatých paketů pro každý rozprostírací faktor 41
7.3 LoRaWAN síť se zapnutým ADR mechanismem - rozložení rozprostíracího faktoru . 43
7.4 LoRaWAN síť se zapnutým ADR mechanismem - DER a počet přijatých paketů pro každý rozprostírací faktor 44

Kapitola 1

Úvod

IoT, nebo-li Internet věcí, v dnešní době zažívá v technologické oblasti jeden z největších rozvkětů. Speciální požadavky pro IoT, přinesly přileżitosti k vzniku nových technologií a jednou z nich je právě technologie LoRaWAN. Inetrnet věcí je velmi široký pojem a spadá do něj několik různých zařízení, pracujících na různých principech. Proto je čtenář z úvodu obeznámen se základními rysy LPWA sítí a proč právě slaví takový úspěch. Po krátkém úvodu do LPWA sítí je vysvětlen pojem LoRa a jak souvisí s technologií LoRaWAN. Načež navazuje kapitola, která obsahuje základní informace o technologii LoRaWAN.

Za cíl si práce dává popsat a zdokumentovat práci se simulační knihovnou FloRa, která slouží k simulování LoRaWAN sítí v prostředí OMNeT++ a vytvořit dvě laboratorní úlohy do odborného předmětu Modelování sítí. Čtenář je postupně seznámen se softwarem OMNeT++ včetně popisu samotného IDE, dále s INET frameworkem, který FloRa využívá, a jejich následnou instalací na operační systémy Windows a Linux Ubuntu. Samotná FloRa je zdokumentována v kapitole 6 společně s jejím přidáním do prostředí OMNeT++.

Závěr práce představují dvě simulace zhotovené za pomoci knihovny FloRa. Obě simulace jsou formou laboratorních úloh a jejich cílem je pozorovat dvě identické LoRaWAN sítě, kde jedna využívá tzv. ADR mechanismus a druhá nikoliv. Data získáné z obou simulací jsou prezentovány především formou grafů a tabulek, z kterých jsou následně vyvozeny závěry a jejich vzájemné porovnání.

Kapitola 2

Low-Power Wide Area Networks

LPWAN jsou sítě o velké rozloze s nízkou energetickou spotřebou. Pokrytí dosahuje až 5km v městské oblasti a až 40km mimo ní. Toho bylo docíleno zvláště díky využití Sub-1GHz pásma, které nabízí robustní i spolehlivou komunikaci, a také pomoci speciálních modulací.

Využitím LPWA technologií se může životnost baterií v zařízeních zvýšit až na 10 let. Toho bylo především docíleno využitím následujících aspektů. Prvním je, že dříve byly hodně používány mesh topologie, aby se zvýšil dosah pokrytí sítí s krátkým dosahem. Cesta k bráně vedla mnohokrát přes několik jiných koncových zařízení, čímž se životnost jejich baterií dost snižovala. Na druhou stranu s využitím LPWA technologií se koncová zařízení připojují přímo ke své bráně a díky snížené komunikaci je možné vypínat nebo zapínat různé hardwarové komponenty. K zvýšení životnosti baterií také pomohlo, že LPWA technologie často využívají MAC protokoly s náhodným přístupem, jako například Aloha, nebo protokoly založené na principu TDMA. A v neposlední řadě, že koncová zařízení přenechávají komplexní úlohy svým bránám.

Úspěch LPWA technologiím přinesly i celkové nízké náklady - cena hardwaru je okolo 5 USD a poplatek za připojení kolem 1 USD. Hlavní roli v tom hraje snížení komplexnosti hardwaru, velké pokrytí brán a používání především bezlicenčního pásma. [1] [2]

Faktory, na které je v LPWA sítích kladen důraz

- Síťová architektura a velká kapacita sítě
- Velký komunikační dosah
- Nízká spotřeba a dlouhá životnost baterie
- Odolnost proti rušení
- Zabezpečení sítě
- Jednosměrná komunikace vs obousměrná komunikace
- Široká paleta aplikací [3]

Kapitola 3

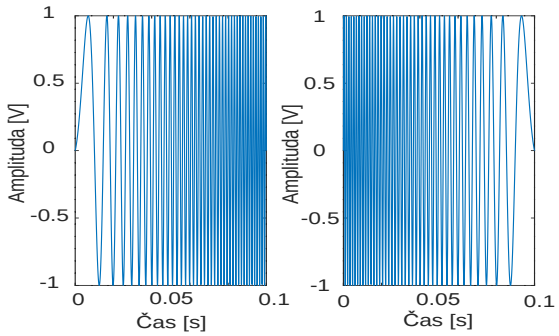
LoRa

LoRa je fyzická vrstva vytvořena společnosti Semtech. Název je odvozen od slovního spojení long-range a využívá se v LoRaWAN technologii. Jako médium pro přenos informací využívá rádiové vlny s dosahem 15km v otevřeném prostoru a až 5km v uzavřeném/městském prostředí. [4]

Modulace je založena na Chirp Spread Spectrum (CSS) technologii, která je dost podobná FSK modulaci v případě úspory spotřeby, ale narozdíl od FSK modulace nabízí daleko větší pokrytí. CSS modulace byla hodně využíváná v armádní a vesmírné komunikaci z důvodů jejího komunikačnímu dosahu a odolnosti proti rušení. [3]

3.1 Modulace

Chirp signál zde slouží jako nosný signál, na který se moduluje zpráva. Má konstantní amplitudu a proměnnou frekvenci. Je tvořen jednotlivými chirp pulzy nesoucí symboly. Pulzy se dělí na dva druhy, první se nazývá up-chirp a nastává pokud se frekvence zvyšuje z nejnižší na nejvyšší. Jestliže se frekvence snižuje z nejvyšší na nejnižší, hovoříme o down-chirp pulzu. Symbol se dále dělí na *"chips,"* nebo-li zakódované bity, kterými lze symbol vyjádřit. Množství těhle bitů určuje rozprostírací faktor (Sreading Factor). [4] [5]



Obrázek 3.1: Up-chirp (vlevo) a down-chirp (vpravo) pulzy

3.2 Spreading Factor

Celkově bylo v Evropě definováno 6 rozprostíracích faktorů - SF7 až SF12. [6] Výhodnou vlastností rozprostíracího faktoru je, že zajišťuje ortogonalitu. To dovoluje přijímači správně detekovat zprávu se SF x, i když se časově překrývá se zprávou mající SF y. Pokud bychom uvažovali fixní šířku pásma, zvýšením rozprostíracího faktoru dosáhneme delší doby trvání symbolu a *"Time on Air"* (ToA), nebo-li času mezi vysláním a přijmutím zprávy. Se vzrůstajícím rozprostíracím faktorem je i citlivost přijímače lepší, viz tabulka 3.1. Na druhou stranu s vyššími hodnotami SF se snižuje přenosová rychlost. Vzhledem k těmto vlastnostem se využívá vyšší rozprostírací faktor pro větší vzdálenosti a naopak nižší pro menší. [4] [5]

Rozprostírací faktor	**Citlivost [dBm]**
7	-123,0
8	-126,0
9	-129,0
10	-132,0
11	-134,5
12	-137,0

Tabulka 3.1: Citlivost pro rozprostírací faktory při šířce pásma 125 KHz [7]

Kapitola 4

LoRaWAN

LoRaWAN definuje komunikační protokol a síťovou architekturu, zatímco LoRa slouží pro vytvoření komunikačního spojení s velkým dosahem. Protokol a síťová architektura má největší vliv na životnost baterie zařízení, kapacitu sítě, QoS, zabezpečení a nabídnutí široké palety aplikací k využití. [3]

LoRaWAN spadá pod neziskovou organizací LoRa-Alliance, která je momentálně největší a nejrychleji rostoucí aliancí v technologickém sektoru s více jak 500 členy. Jednotliví členové mezi sebou spolupracují a vyměňují si své zkušenosti. [8]

Využívané frekvenční pásmo je v Evropě 867-869 MHz s šířkou pásma kanálu pro uplink 125/250 KHz a pro downlink 125 KHz. V Americe se využívá frekvenční pásmo 902-928 MHz, kde šířka pásma kanálu je pro uplink 125/500 KHz, zatímco pro downlink 500KHz. V Číně jsou využívány frekvence 470-510 MHz. V Koreji a Japonsku 920-925 MHz. V Indii pak 865-867 MHz.

LoRaWAN definuje v Evropě celkem 10 kanálů. 8 z nich je určených pro vícenásobnou datovou rychlost od 250 bps do 5,5 kbps, jeden pro vysokorychlostní datovou rychlost s rychlostí 11 kbps a samotný FSK kanál dosahující rychlosti 50 kbps. [3] Tabulka 4.1 obsahuje jednotlivé přenosové rychlosti, jejich konfigurace a tomu odpovídající bit-rate.

DataRate	**Konfigurace**	**Bit-rate [bit/s]**
0	LoRa: SF12 / B:125 KHz	250
1	LoRa: SF11 / B:125 KHz	440
2	LoRa: SF10 / B:125 KHZ	980
3	LoRa: SF9 / B:125 KHz	1760
4	LoRa: SF8 / B:125 KHz	3125
5	LoRa: SF7 / B:125 KHz	5470
6	LoRa: SF7 / B:250 KHz	11000
7	FSK: 50 kbps	50000

Tabulka 4.1: EU863-870 DataRate [6]

4.1 Síťová architektura

LoRaWAN využívá hvězdicovou topologii, kde koncové zařízení je připojeno k jedné nebo více brán.

Architektura se typicky skládá ze čtyř částí - koncové zařízení, koncentrátor (gateway), řídící server a aplikační server.

- **Koncové zařízení** - Je například senzor připojený k LoRaWAN koncentrátoru, který může sloužit k snímaní fyzikálních podmínek nebo enviromentálních událostí.

- **Koncentrátor** - Přeposílá přijímané LoRa modulované zprávy z koncového zařízení do síťového serveru, s kterým komunikuje využitím IP protokolu. Jestliže má koncové zařízení v dosahu více koncentrátorů, posílá zpávy všem. Duplicitu zpráv řeší síťový server. Ten nechává zprávu s nejlepším RSSI.

- **Řídící server** - Se stárá o chod celé sítě. Například dynamicky kontroluje síťové parametry, vytváří zabezpečenou 128 bitovou AES komunikaci mezi koncovým zařízením a aplikací uživatele, nebo kontroluje provoz.

- **Aplikační server** - Je zodpovědný za bezpečné zpracování, správu a interpretaci dat z koncového zařízení uživateli. [4]

4.2 Třídy

Z důvodu velkého množství aplikací a rozdílných požadavků, LoRaWAN rozděluje koncová zařízení do tří tříd. Podle latence downlinku z koncentrátoru a životnosti baterie. [3]

4.2.1 Třída A

Jedná se o energeticky nejúspornější třídu, kterou musí podporovat každé LoRaWAN koncové zařízení. Komunikace je vždy iniciována koncovým zařízením - kdykoliv může zaslat data, které jsou doprovázena dvěma krátkými časovými intervaly určenými pro příjem dat.

4.2.2 Třída B

Má navíc oproti třídě A další přijímací okna v nastavenou dobu. Ty se otevírají na základě v čase synchronizované beacon zprávy. To poskytuje síti schopnost odesílat downlink zprávy s deterministickou latencí. Ta je programovatelná a může být až 128 sekund, aby vyhovovala různým aplikacím. Proto se stále hodí pro baterie napájené aplikace.

4.2.3 Třída C

Zařízení třídy C mají největší energetickou spotřebu, protože kromě doby určené pro uplink je jejich přijímací okno neustále otevřené. Tudíž se už nehodí pro baterii napájené aplikace, ale spíš pro aplikace s trvalým napájením. [9]

4.3 Regulace

K frekvenčnímu pásmu 867-869 MHz se nesou omezení stanovené ETSI. Mezi ně patří regulace fyzického média, nebo-li maximální doba, kdy vysílač zařízení může být zapnutý, nebo také maximální čas, ve kterém zařízení můžu vysílat v rámci jedné hodiny. První možností, jak vyhovět regulacím stanovené ETSI je implementování Listen Before Talk Adaptive Frequency Agillity (LBT AFA). Protokol LoRaWAN využívá druhou možnost v podobě duty-cycle. [6]

4.3.1 Duty-cycle

Duty-cycle, nebo také klíčovací poměr, vyjádřuje kolik procent času z nějakého celkového času dané zařízení vysílalo. Jestliže zařízení vysílalo po dobu dvou jednotek času z celkových deseti jednotek času, znamená to, že duty-cycle je 20%

Obrázek 4.1: Duty-cycle 20% [10]

V případě, že by zařízení vysílalo na více kanálech, jednotlivé hodnoty duty-cycle se sčítají. Vezme-li se předchozí případ a namísto něho by zařízení vysílalo na třech kanálech se stejnou hodnotou duty-cycle, tak celková hodnota klíčovacího poměru by činila 60%.

Použijou-li se sub-pásma při stejném počtu kanálu a stejnou hodnotou duty-cycle, kdy jeden kanál je v jednom sub-pásmu a zbylé dva v jiném sub-pásmu, tak by hodnota prvního sub-pásma činila 20% a druhého 40%. Celková hodnota klíčovacího poměru pro zařízení je stále 60%, ale nyní se rozložila do dvou sub-pásem. [10]

Podle dokumentu VO-R/10/12.2019-9 vydaného českým telekomunikačním úřadem, spadá LoRa do pásma *"g,"* kde je klíčovací poměr 0,1%, který platí pro koncová zařízení i koncentrátory. [11]

4.4 Formát zprávy

- **Fyzická vrstva** - Terminologie LoRa rozlišuje mezi zprávami určené pro uplink a downlink. Obě zprávy obsahují preambuli, fyzickou hlavičku LoRa (PHDR), hlavičku CRC (PHDR_CRC) a užitečná data (PHYPayload). Uplink zpráva má ještě navíc pole CRC, které zajišťuje integritu užitečných dat.

- **MAC vrstva** - První oktet v PHYPayload je MAC hlavička (MHDR), nesoucí informace o typu zprávy a podle které hlavní verze byl rámec zakódován. Ta je následováná datovým rámcem (MACPayload) a kódem integrity zprávy (MIC). Datový rámec obsahuje hlavičku rámce (FHDR), která v sobě nese informace ohledně adresy koncového zařízení, oktet *"frame control"* (FCtrl), dva oktety *"frame counter"* (FCnt) a až 15 oktetů *"frame options"* (FOpts) sloužící k transportu MAC příkazů. Dále datový rámec nese informace o použitém portu (FPort) a pole užitečných dat (FRMPayload).

  - **FPOrt** - Může nabývat hodnot 0-224.
    - Pokud port není přítomný, nebo není 0, přenáší se MAC příkazy ve *"frame options."*
    - V případě portu 0 se ve FRMPayload přenáší pouze MAC příkazy.
    - Porty 1-223 jsou pro konkrétní aplikace.
    - Port 224 byl přidělen testovacímu protokolu MAC vrstvy. [12]

Obrázek 4.2: Formát zprávy

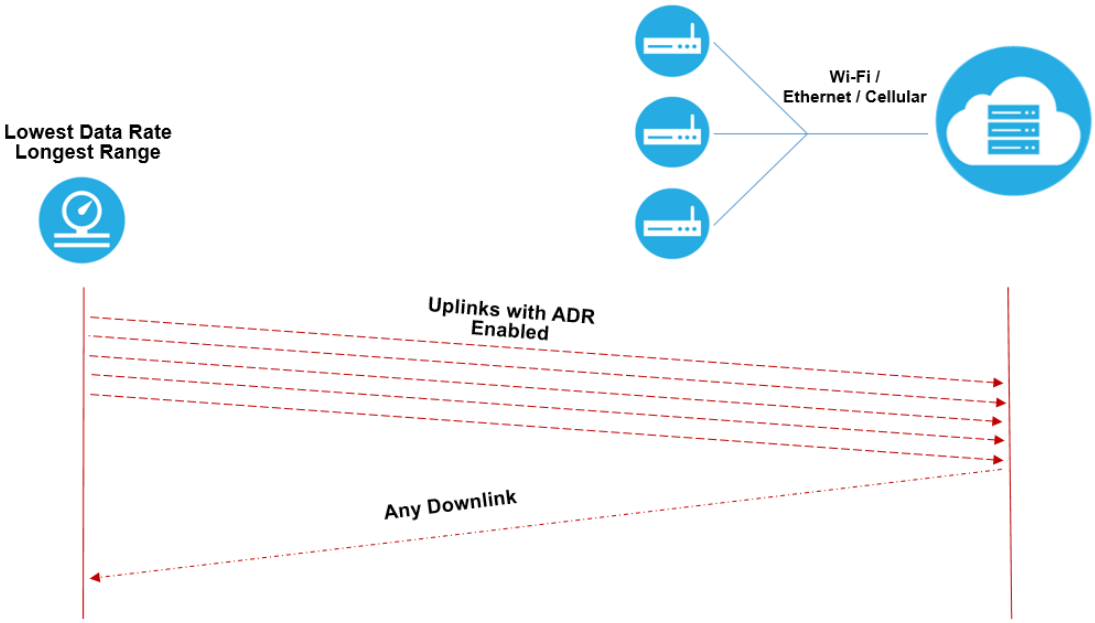
4.5 Adaptive Data Rate

Je mechanismus pro zvolení patřičné přenosové rychlosti na základě parametru link budget. Příkladem může být zařízení umístěné blízko koncentrátoru, pro které je tedy zbytečné použít SF12, mající velký link budget. Namísto toho se hodí využít menší rozprostírací faktor.

ADR mechanismus může být zahájen ze sítě, nebo koncového zařízení. Nicméně zda ho použít, nebo nepoužít rozhoduje aplikace koncového zařízení. Pokud aplikace rozhodne, že ano, vloží se do

hlavičky rámce *"ADR bit"* v poli *"frame control"* (FCtrl), čímž se sděluje sítí, že může koncovému zařízení určit jeho přenosovou rychlost.

Tahle zpráva s nastaveným *"ADR bitem"* následně putuje přes všechny dostupné koncentrátory do síťového serveru. Ten čeká, dokud nenashromáždí více výsledků a jakmile jich má dostatek, vypočítá medián, určí link budget a nejvyšší podporovanou přenosovou rychlost. Po té je serverem zaslán MAC příkaz patřičnému zařízení k změně přenosové rychlosti. Koncové zařízení změní SF a všechny budoucí uplink zprávy jsou nastavené na tuhle přenosovou rychlost.



Obrázek 4.3: ADR mechanismus [7]

Při situaci, že by se spojení nedokázalo navázat, se v koncovém zařízení inkrementuje tzv. *"ADR acknowledge counter"* při každé odeslané zprávě, do doby než dosáhne definovaného maxima. Jakmile dojde k maximu *("ADR acknowledgement limit,")* nastaví tzv. *"ADR acknowledge request bit"* v hlavičce rámce do pole *"frame control"* (FCtrl) a ten se pošle do sítě.

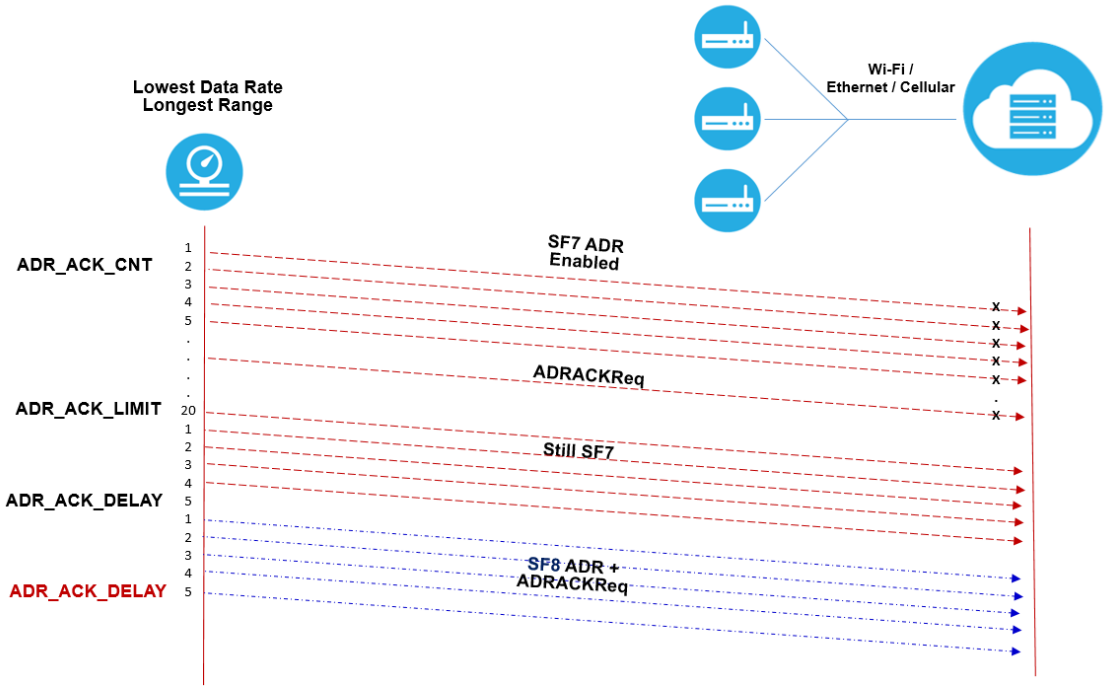
FCtrl:

Bit #	7	6	5	4	[3:0]
FCtrl bits	ADR	ADRACKReq	ACK	Fpending	FOptsLen

Obrázek 4.4: Pole frame control

Dále čeká na odpověď definovanou dobu známou jako *"ADR acknowledge delay,"* protože server může vyhodnocovat ještě jiné další požadavky. *"ADR acknowledge delay"* není definovaný v čase, ale jedná se o počet odeslaných zpráv. Důvodem je zachování nízkých nákladů a nízké energetické spotřeby koncového zařízení. Jestliže nepříjde žádná odpověď a *"ADR acknowledge delay"* dosáhne

svého maxima, zařízení sníží přenosovou rychlost (zvýší SF) a pokračuje s novým rozprostíracím faktorem v posílání zpráv s požadavkem na potvrzení ADR. Když je potvrzení přijato, zařízení použije novou serverem definovanou přenosovou rychlost, dokud neobdrží instrukce, aby ji znovu změnil prostřednicvtím ADR mechanismu. Jestli nastane situace, že i po zvýšení SF zařízení nedostane odpověď, tak jej zvedá do doby, než obdrží odpověď, nebo do okamžiku kdy dosáhne nejvyššího rozprostíracího faktoru. [7]



Obrázek 4.5: Neúspěšný ADR mechanismus [7]

4.6 Řešení kolizí

Síť protokolu LoRaWAN funguje na bázi protokolu Aloha. [3]

4.6.1 Aloha

Je protokol s náhodným přístupem vyvinuty na universitě v Havaji na začátku 70. let. Hlavní úlohou tohoto protokolu je určení, které zařízení dostane jako další příležitost vysílat.

Protokol lze definovat pomoci následujících pravidel:

- Jakkákoliv stanice může začít vysílat kdykoliv.
- Provoz na sítí se nijak nesnímá.

- Můžou nastat kolize.

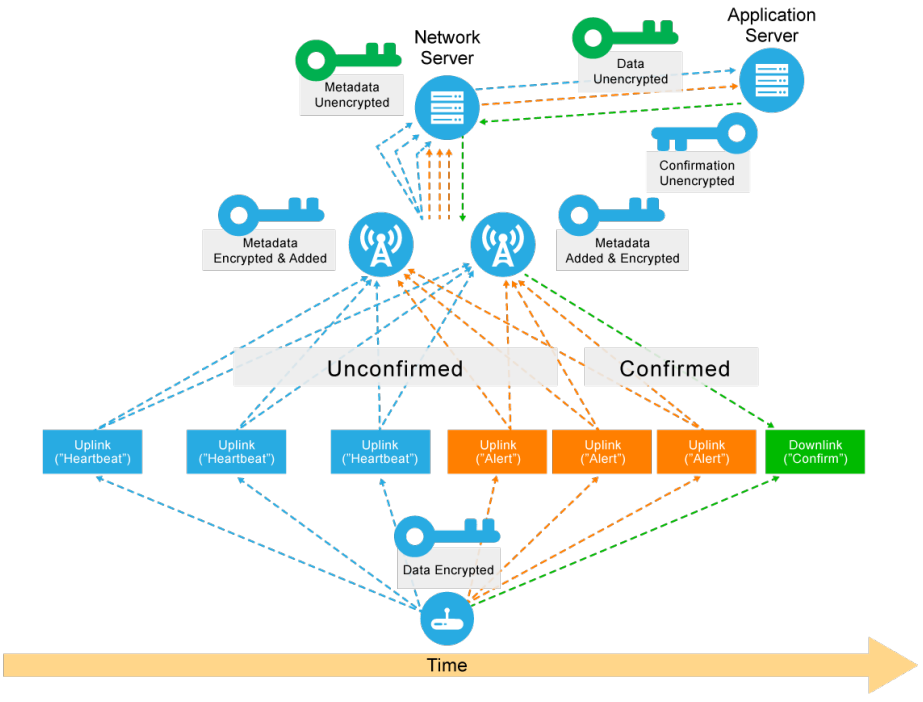
- Protokol funguje na bázi potvrzování, a proto není potřeba žádný mechanismus pro detekci kolizí.

- Stanice může znovu vysílat až po uplynutí náhodného času. [13]

4.6.2 Potvrzovací zprávy v protokolu LoRaWAN

Koncové zařízení při zaslání zprávy si může, ale taky nemusí vyžádat její potvrzení.

Příkladem může být obrázek 4.6, kde jako koncové zařízení vystupuje detektor kouře. Ten vysílá do sítě periodické zprávy (modrý přenos), sloužící jen k potvrzení, že dané zařízení funguje. Jakmile detektor něco zaznamená, začne vysílat zprávy, u kterých vyžaduje potvrzení (oranžový přenos), do doby, dokud nebudou potvrzeny (zelený přenos). [14] Dalším příkladem potvrzovacích zpráv je již zmíněný ADR mechanismus.



Obrázek 4.6: Mechanismus pro potvrzování zprávy [14]

4.7 Zabezpečení

Zabezpečení tvoří dva 128 bitové šifrovací klíče - síťový (NwkSkey) a aplikační (AppSkey) využívající 128 AES algoritmus.

Síťový se sdílí mezi koncovým zařízením a síťovým serverem. Šifruje přenášenou užitečnou informaci a veškeré rádiové informace pro zajištění optimálního radiového přenosu. Jeho hlavním účelem je, aby do sítě nebyly vpuštěny neznámé koncové zařízení.

Aplikační se sdílí mezi oběma konci, šifruje užitečnou informaci a chrání privátnost dat určené jen uživateli.

Existují dvě možnosti, jak dostat klíče do zařízení a následně je pomoci nich aktivovat v sítí.

- **ABP** (Activation by personalization) - Klíče se vygenerují před výrobou koncového zařízení a následně se do něj nahrají.
- **OTAA** (Over-the-air activation) - Spočívá v nahrání dočasných klíčů, které slouží k prvotní autorizaci a po první odeslané zprávě je zařízení přidělen aplikační klíč z aplikačního serveru a síťový ze síťového serveru. [15]

Kapitola 5

OMNeT++ a INET framework

OMNeT++ (**O**bjective **M**odular **N**etwork **T**estbed in C++) je objektově orientovaný, modulární framework pro simulování sítí, pracující na principu diskrétních událostí. Má obecnou architekturu, takže jej lze použít pro řešení různých situací jako například:

- modelování drátových a bezdrátových komunikačních sítí
- modelování protokolů
- modelování multiprocesorů a dalších distribuovaných hardwarových systémů
- ověřování hardwarových architektur
- a všeobecně pro modelování jakýchkoliv systémů, pro které se hodí využití diskrétních modulací a lze je pohodlně mapovat na entity komunikující výměnou zpráv

OMNeT++ je sám o sobě spíš jakási infrastruktura, která poskytuje nástroje k vytváření simulací. Základem téhle infrastruktury je komponentní architektura pro simulační modely, které jsou poskládány ze znovu použitelných komponentů zvané moduly.

5.1 Diskrétní události

Narozdíl od kontinuálních systémů, kde jsou změny kontinuální, systémy diskrétních událostí jsou systémy, kde ke změnám stavů (událostem) dochází v diskrétních případech v čase a událostem trvá nulový čas k provedení. Předpokládá se, že se nic zajímavého nestane mezi dvěma po sobě jdoucími událostmi, to znamená, že v systému mezi událostmi nedojde k žádné změně stavu.

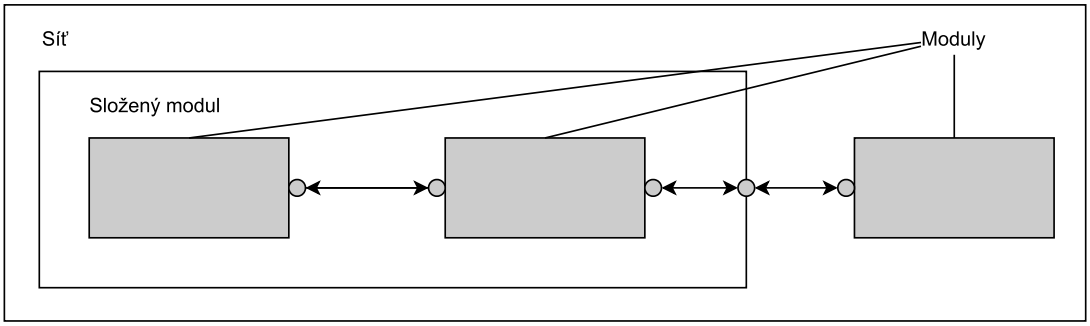
Například počítačové sítě jsou obvykle považovány za systémy diskrétních událostí. Jedná se o události jako například:

- záhájení přenosu paketů

- ukončení přenosu paketů
- vypršení časového limitu opakovaného přenosu

5.2 Moduly

Moduly můžou být mezi sebou propojeny skrz brány/porty a kombinovány do složených modulů. Komunikují prostřednictvím předávání zpráv, kde zprávy mohou nést libovolné datové struktury. Zprávy mohou předávat podél předdefinovaných cest prostřednictvím bran a spojení, nebo přímo do svého cíle. Co se parametrů modulů týká, lze je použít k přizpůsobení chování modulu a/nebo k parametrizaci topologie modelu. Moduly na nejnižší úrovni hierarchie modulů se nazývají jednoduché moduly a zapouzdřují chování modelu. Jednoduché moduly jsou programovány v C++ a využívají simulační knihovnu.



Obrázek 5.1: Stavba modulů

5.3 Stavba OMNeT++

OMNeT++ se skládá z následující částí:

- Popis(y) topologie jazyka NED (soubory .ned), které popisují strukturu modulu s parametry, branami, atd.
- Definice zpráv (soubory .msg), které umožňují definovat typy zpráv a přidávat k nim datová pole. OMNeT++ pak tyhle definice zpráv překládá do plnohodnotných C++ tříd.
- Jednoduché zdroje modulů - soubory C++ s příponou .h/.cc.

Simulační systém poskytuje následující komponenty:

- Simulační kernel, který je napsán v C++ a zkompilován do sdílené nebo statické knihovny.

- Uživatelská rozhraní - ty jsou použity pro provádění simulací, usnadnění ladění, nebo dávkové provádění simulací. Jsou psány v C++, kompilovány do knihoven.

Simulační programy jsou sestaveny z výše uvedených komponent. Nejprve jsou soubory .msg přeloženy do C++ využitím opp_msgc. programu. Poté jsou všechny C++ zdroje zkompilovány a propojeny se simulačním kernelem a knihovnou uživatelského rozhraní za účelem vytvoření spustitelného souboru simulace nebo sdílené knihovny. Při spuštění simulačního programu se soubory NED načítají dynamicky ve svých původních textových formách.

Simulaci lze zkompilovat jako samostatný spustitelný program, nebo jako sdílenou knihovnu, která běží pomocí OMNeT++ nástroje zvaný op_run. Při spuštění programu jsou jako první čteny NED soubory a následně konfigurační s příponou .ini. Tyhle konfigurační soubory obsahují řídící nastavení, která řídí, jak je simulace spouštěná, parametry modulů, atd.

Výstup simulace je obsažen ve výsledkových souborech, mezi které patří výstupní vektorové soubory, výstupní skalární soubory, případně výstupní soubory definované uživatelem. OMNeT++ IDE poskytuje bohaté prostředí pro analýzu těchto výstupních souborů. Je také možné, je zpracovat pomocí různých nástrojů a programovacích jazyků včetně Matlabu, GNU R, Perlu, Pythoun a spreadsheet programů, díky tomu, že jsou textové, řádkově orientované. [16]

5.4 OMNeT++ IDE

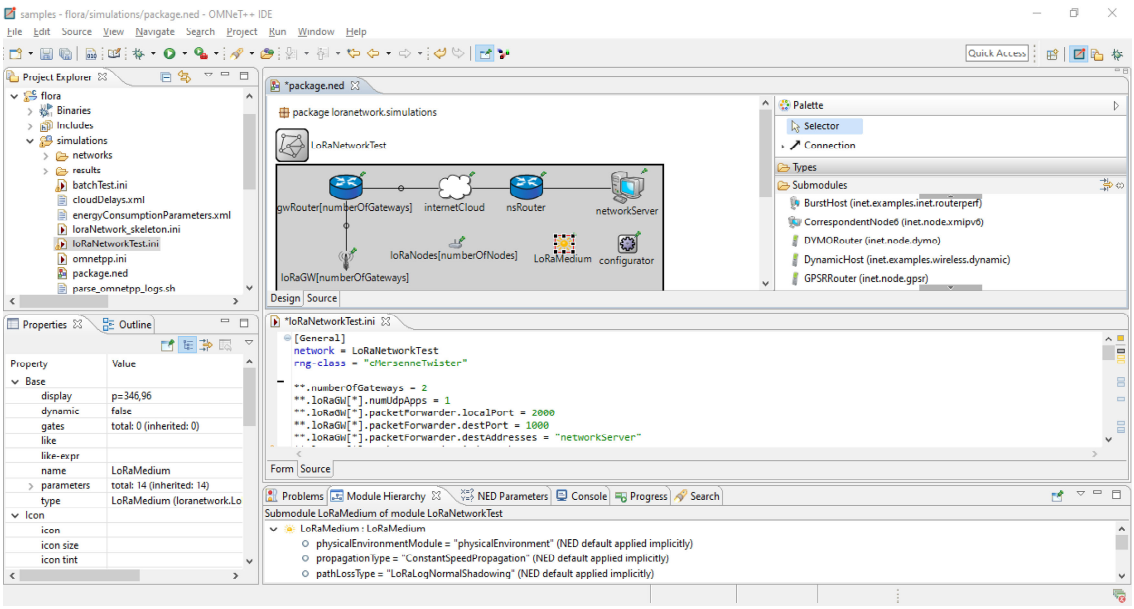
OMNeT++ IDE lze vidět na obrázku 5.2. Skládá se z horní lišty obsahující několik záložek, například záložku *"File,"* kde najdete jednotlivé nabídky, jako třeba vytváření, ukládání, načítaní projektů, souborů atd., nebo *"Search"* pro vyhledávání v projektech. Užitečná je i záložka *"Window,"* kde si můžete upravit vzhled IDE. Pod touhle lištou pak najdete prostředky pro sestavení projektů, jejich spuštění, či funkce *"Undo"* a *"Redo"*.

Po levé straně se nachází *"Project Explorer,"* obsahující naimportované projekty včetně jejich obsahu, dále záložka *"Outline,"* která poskytuje přehled souboru a záložka *"Properties."* Ta zobrazuje vlastnosti vámi nakliknutého modulu. Na samotném spodu IDE je okno určené pro konzoli, výpis chyb, atd.

5.4.1 NED soubory

Uprostřed pracovní plochy jde vidět .ned soubor a v něm moduly, které tvoří LoRaWAN síť. U tohohle typu souboru můžete překlikávat mezi grafickým zobrazením a zdrojovým kódem, využitím tlačítek *"Design"* a *"Source,"* které lze vidět na spodu .ned souboru. Pokud zrovna operujete v grafickém zobrazení, můžete využít nabídky *"Palette,"* kde jsou tlačítka pro vybrání modulu, či k jeho propojení s jiným využitím tlačítka *"Connection,"* a nebo nabídku *"Submodels"* obsahující dostupné moduly. V téhle nabídce stačí na vámi vybraný model kliknout, poté kliknout do vaší sítě

na vámi určené místo a modul se zde přidá. Vámi přidané moduly se zobrazí i ve zdrojovém kódu, kde jim můžete například přidat nějaké parametry, nebo změnit název.



Obrázek 5.2: OMNeT++ IDE

5.4.2 INI soubory

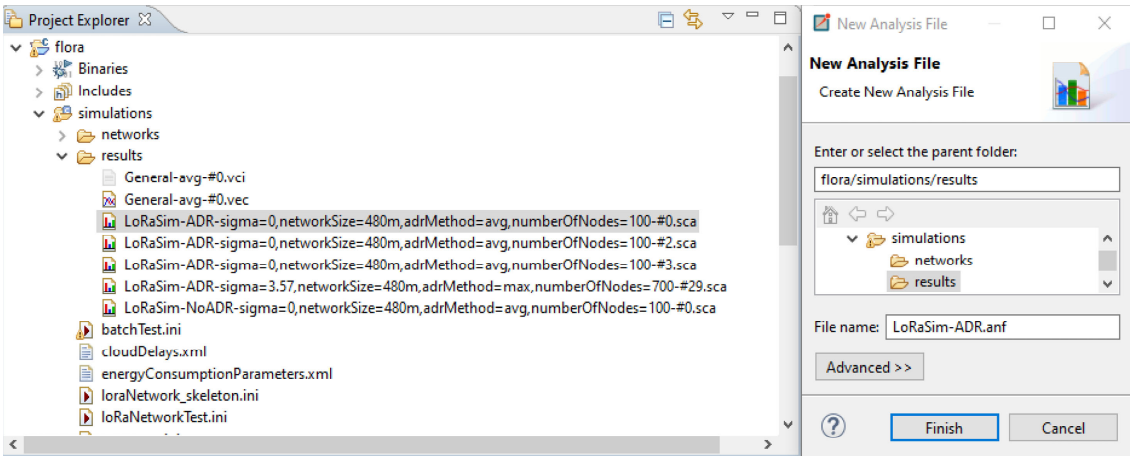
Pod .ned souborem je zobrazen .ini soubor, který slouží, jak už bylo zmíněno, ke konfiguraci. I tyhle soubory nabízí dvě zobrazení. Na obrázku 5.2 lze vidět zobrazení v *"Source"* režimu určený ke konfiguraci parametrů sítě/modulů atp. INI soubory můžete zobrazit i v režimu *"Form."* Tenhle režim poskytuje možnost měnit nejrůznější nastavení, či samotné parametry dané simulace.

5.4.3 Výsledkové soubory

Výsledkové soubory obsahují výsledky dokončené simulace. Obvykle se nachází v složce *"simulations"* v podsložce *"results"* (levá část obrázku 5.3). Dvojklikem levým tlačítkem myši je lze převést do .anf souboru a umístit ho, kam chcete (pravá část obrázku 5.3).

Výsledný .anf soubor můžete vidět na obrázku 5.4. V něm můžete přepínat mezi třemi režimy, z nichž jsou data k naleznutí v *"Browse Data."* Pokud máte víc náměrů v rámci jedné simulace, všechny se promítnou.

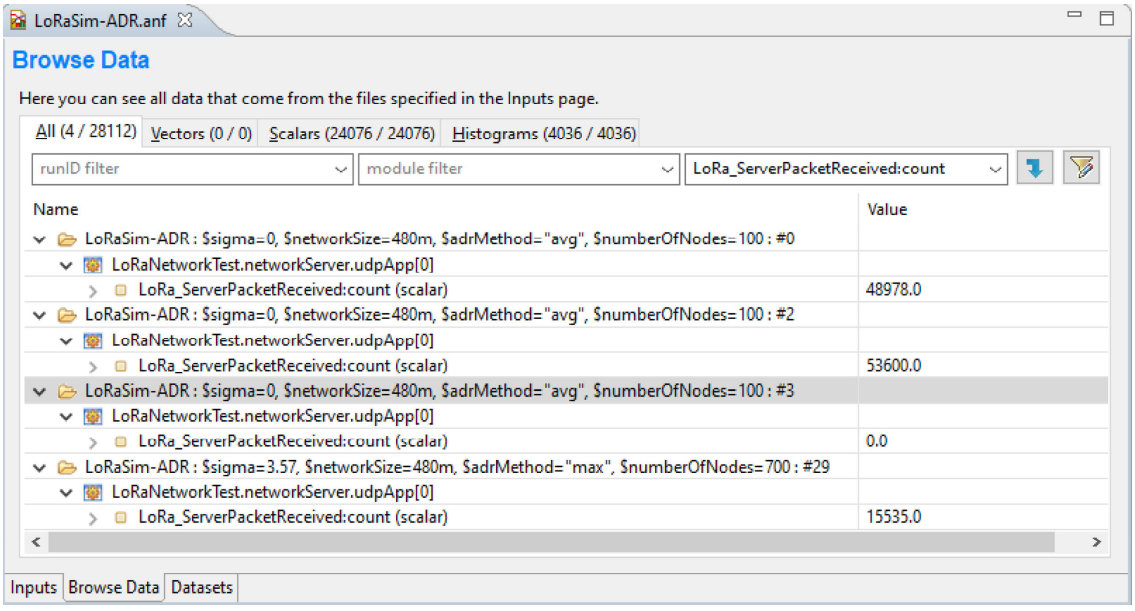
Na obrázku jsou celkově 4 náměry, každý takový náměr má svou složku, kterou lze rozkliknout. V každé složce se nachází všechny vámi použité modely a data, která zaznamenaly. Jednotlivá data můžou být filtrována. K tomu slouží horní filtrovací nabídka. První nabídka je určena pro zvolení náměru, druhá pro modul, či model a třetí pro konkrétní hodnotu. V obrázku 5.4 jsou použity



Obrázek 5.3: Výsledkové soubory a jejich převod do .anf souboru

následující filtry: zobrazení všech náměrů, modul *"udpApp"* v síťovém serveru a konečný počet LoRa paketů, který přijal.

Data jdou i exportovat do souborů různých formátů včetně použitých filtrů. Pro export klikněte pravým na vámi vybraný náměr, přejděte na *"Export Data"* a zvolte formát, do kterého chcete exportovat. Po kliknutí se vám zobrazí nabídka, kde si můžete určit cílový soubor, popřípadě přizpůsobit různá nastavení.



Obrázek 5.4: Výsledný .anf soubor s filtrem

5.5 INET Framework

INET Framework je open-source knihovna modelů pro simulační prostředí OMNeT++. Poskytuje protokoly, agenty a další modely pro vědecké pracovníky a studenty pracující s komunikačními sítěmi.

INET například obsahuje modely pro TCP, UDP, IPv4, IPv6, OSPF, BGP atd., drátové a bezdrátové linkové protokoly jako Ethernet, PPP, IEEE 802.11 atd. Dále podporuje protokoly jako například MANET, DiffServ, MPLS se signalizací LDP a RSVP-TE, několik aplikačních modelů a mnoho dalších protokolů a komponentů.

Několik dalších simulačních frameworků bere INET jako základní kámen a rozšiřuje jej do konkrétních směrů, jako jsou například automobilové sítě, overlay/peer-to-peer sítě, nebo LTE. [17]

5.6 Instalace OMNeT++

Mezi podporované platformy OMNeT++ patří:

- Windows 7 a Windows 10 / 64-bit
- MacOS 10.12
- Linuxové distribuce
  - Ubuntu 16.04 LTS
  - Fedora Core 25
  - Red Hat Enterprise Linux Desktop Workstation 7.x
  - OpenSUSE 42

IDE je pak podporováno na:

- Linux x86; 64-bit
- Windows 7, 10; 64-bit
- MacOS 10.12

Nicméně simulace samy o sobě, bez podpory IDE můžou běžet i na unixových distribucích podporující C++ compiler. [18]

5.6.1 Zprovoznění OMNeT++ pro Linux a Windows

V téhle sekci se podíváme na postup pro instalaci softwaru OMNeT++. Mnou testované platformy byly Windows 10 a Linux Ubuntu 16.04 LTS. Pro obě platformy platí víceméně stejný postup, jen

v případě instalace OMNeT++ pro Linux je ještě potřeba nainstalovat různé balíčky. Jaké a jak je nainstalovat můžete najít v následujícím zdroji: [18].

Pro naše účely budeme potřebovat OMNeT++ verzi 5.2.1., kterou lze nalézt na oficiálních OMNeT++ webových stránkách: `https://omnetpp.org/download/old`. U každé verze je uvedeno datum vydání, tlačítko *"read more"*, sloužící pro zobrazení novinek v dané verzi a spodní lišta s jednotlivými operačními systémy. Při naléznutí verze 5.2.1. zvolte ve spodní liště vámi preferovaný operační systém a stáhněte tlačítkem *"download."* Jakmile se dokončí stahování souboru, rozbalte jej do vámi vybrané složky (při instalaci v operačním systému Windows je doporučeno se vyhýbat složkám obsahující mezery v názvu, jako například *Program Files*).

Nyní přejděte do rozbalené složky využitím terminálu, otevřete v ní soubor *configure.user*, upravte řádek **PREFER_CLANG=yes** na **PREFER_CLANG=no** a soubor uložte. V případě instalace v operačním systému Windows je nutné spustit MinGW konzoli, nacházející se v rozbalené OMNeT++ složce. Pro to využijte následující příkaz v terminálu:

$ mingwenv.cmd

a potvrďte stisknutím libovolného tlačítka. Po stisknutí se zahájí extrahování potřebných souborů. Až se extrahování dokončí, stiskněte ještě jednou libovolnou klávesu a to vás přesměruje do MinGW konzole.

V tomhle bodě se instalace pro Ubuntu a Windows opět sbíhají. V případě Windowsu se nacházíte v MinGW konzoli a v Ubuntu v terminálu v rozbalené OMNeT++ složce. Jak do konzole, tak i terminálu zadejte tyhle příkazy:

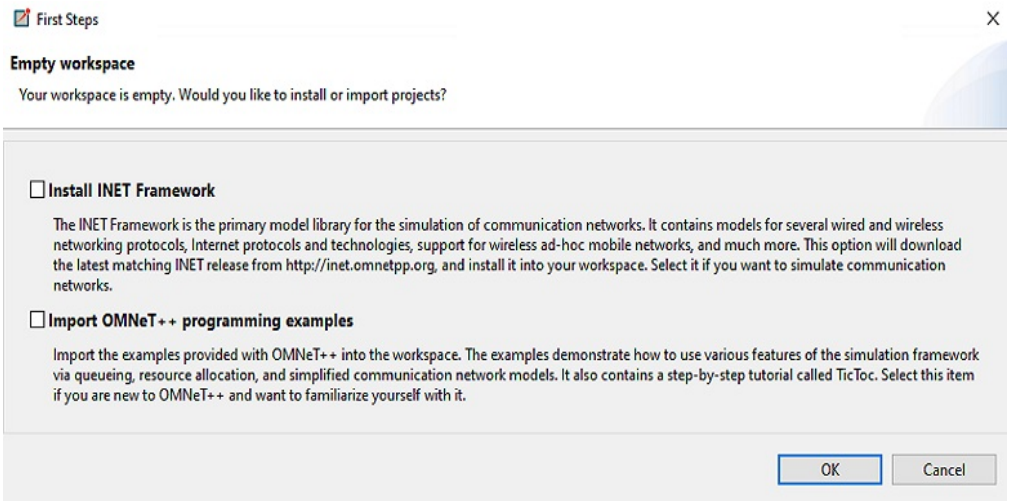
$ ./configure

$ make

Jakmile oba příkazy dojedou, můžete spustit OMNeT++ příkazem:

$ omnetpp

Při prvním spuštění softwaru OMNeT++ se zobrazí uvítací stránka, kterou odkřížkujte. Následně se ukáže nabídka s volbou nainstalování INET frameworku a vložení předem připravených ukázek. Políčko pro nainstalování INETu nechte prázdné, protože budeme potřebovat vložit starší verzi. Druhé políčko závisí na vaší volbě.



Obrázek 5.5: Nabídka pro instalaci INET Framework

5.6.2 Přidání INET framework do OMNeT++

Zprovoznění INET frameworku je pro oba operační systémy stejný. Pro naše účely využijeme verzi 3.6.3, která je dostupná na `https://inet.omnetpp.org/2017-12-22-INET-3.6.3-released.html`. Stáhněte soubor a extrahujte ho do složky vaší volby. V OMNeT++ IDE postupujte následovně - *File -> Import* -> rozbalte nabídku *General -> Existing Projects to the Workspace* -> v *Select root directory* přejděte do složky, kde se INET nachází a zvolte ho. Ten se zobrazí nalevo v simulacích. Následně zvolte buď možnost *Project -> Build*, nebo *Ctrl+B* pro sestavení. Po sestavení by mělo být vše připravené k práci.

Kapitola 6

FloRa Framework

FloRa je open-source framework pro realizaci LoRa simulací. Běží v softwaru OMNeT++, konkrétně ve verzi 5.2.1 a společně s ním využívá INET framework verzi 3.6.3. FloRa implementuje LoRa fyzickou vrstvu, LoRaWAN MAC protokol, podporuje obousměrnou komunikaci a poskytuje end-to-end simulace včetně backhaul sítě.[19]

6.1 Základní informace

Co se fyzické vrstvy týká, FloRa nabízí konfiguraci rozprostíracího faktoru, středovou frekvenci, šířku pásma, code rate a přenosový výkon. Na základě těhle parametrů se ve Floře stanovuje komunikační dosah a pravděpodobnost výskytu kolize.

Přijímaný výkon závisí na útlumu a ztrátach signálu skrz překážky a pokud je větší než citlivost přijímače, je přenos úspěšný. Tuhle závislost modeluje tzv. *"log-distance path loss model with shadowing,"* jehož výsledkem jsou ztráty při přenosu [dB] na základě vzdálenosti mezi přijímačem a vysílačem, a jenž má následující tvar:

$$L(d) = L(d_0) + 10nlog(\frac{d}{d_0}) + X_\sigma$$

kde:

- $L(d_0)$ - ztráty ve vzdálenosti d_0 [dB]
- n - exponent ztrát
- X_σ - značí nulový průměr Gaussova rozdělení náhodné proměnné se standardní derivací σ

Dále FloRa podporuje městské i příměstské prostředí s rozdílnými parametry pro ně, které jsou zobrazené v tabulce 6.1.

Kolize fungují na principu, že dva LoRa přenosy spolu nekolidují, pokud mají rozdílný rozprostírací faktor. Popřípadě jestli mají dva signály stejný rozprostírací faktor, může být silnější signál

Prostředí	(d_0) **[m]**	$L(d_0)$ **[dB]**	**n**	σ **[dB]**
Městské	40	127,41	2,08	3,57
Příměstské	1000	128,95	2,32	7,08

Tabulka 6.1: FloRa parametry pro prostředí

dekódován, pokud výkon obou signálu se liší v 6 dBm a aspoň 5 symbolů z preambule z daného rámce bylo detekováno.

6.2 Spotřeba elektrické energie

Energetický výdej koncových zařízení je modelován pomoci *"energy consumer module,"* kde se spotřebovaná energie odvíjí od času stráveného v patřičném režimu. Celkově jsou tři hlavní režimy - vysílací, přijímací a spací. Do spacího režimu dané zařízení přechází po příjmutí, nebo odeslání zprávy. Spotřebovaná energie ve vysílacím režimu závisí na úrovni přenosového výkonu. Hodnoty okamžitého proudu pro každou úroveň přenosového výkonu jsou získány z [20]. Proud čerpaný během přijímacího režimu a režimu spánku jsou odvozeny z Semtech datasheetu SX1272/73 s napájecím napětím 3,3V

6.3 Síťová architektura

FloRa podporuje všechny základní zařízení. Koncentrátorů může být v rámci celé sítě několik, ty podporují LoRa přenosy zároveň na několika kanálech (v souladu s LoRaWAN specifikacemi) mezi nimi a koncovými zařízeními. Se síťovým serverem jsou kocentrátory propojeny přes IP protokol využitím INET frameworku, například Ethernetem, nebo pomocí Wi-Fi spojení. Jelikož je možné použít více koncentrátorů a jedno koncové zařízení jich může mít v dosahu víc, jsou síťové servery vybaveny filtrováním duplikovaných paketů. Také jsou schopny rozhodnout o zaslání dat přes koncentrátor, s kterým mají nejlepší spojení. V rámci FloRy je možné využívat i ADR mechanismus. Ten lze zapnout v koncovém zařízení (ADR-NODE) - 1 i síťovém serveru (ADR-NET) - 2. Vývojáři FloRy uvádějí, že ADR-NET je implementován na základě toho běžícího v The Things Network, který je založen na referečním algoritmu Semtech. Ten je podrobněji popsaný v dřívější sekci 4.5.

6.4 ADR+

Vývojáři dále představují nový algoritmus zvaný ADR+. Ten mění ADR-NET v prvním řádku, kde funkce *"max"* je nahrazená funkcí *"average."* K změně se rozhodli z důvodu, že použití maximální hodnoty SNR k odhadu kvality spojení je vhodné pouze za ideálních podmínek. [21]

Algorithm 1 ADR-NODE

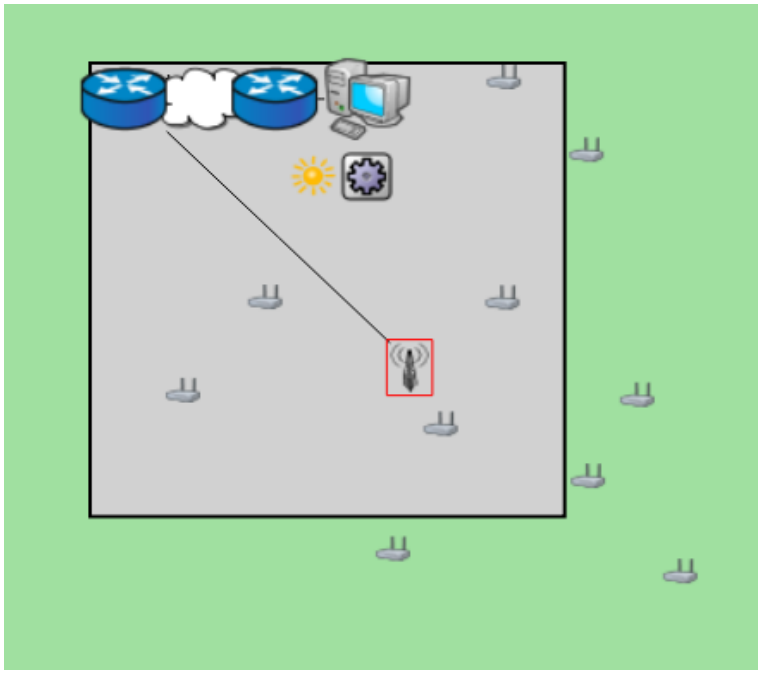
$ADR\_ACK\_LIMIT \leftarrow 64$
$ADR\_ACK\_DELAY \leftarrow 32$
$ADR\_ACK\_CNT \leftarrow 0$
if uplink transmission **then**
    $ADR\_ACK\_CNT \leftarrow ADR\_ACK\_CNT + 1$
if $ADR\_ACK\_CNT == ADR\_ACK\_LIMIT$ **then**
    Request response from network server
if $ADR\_ACK\_CNT \geq ADR\_ACK\_LIMIT + ADR\_ACK\_DELAY$ **then**
    increase SF
if downlink transmission received **then**
    $ADR\_ACK\_CNT \leftarrow 0$

Algorithm 2 ADR-NET

$SNR_m \leftarrow$ max(SNR of last 20 frames)
$SNR_{req} \leftarrow$ demodulation floor(current data rate)
$deviceMargin \leftarrow 10$
$SNR_{margin} \leftarrow (SNR_m - SNR_{req} -$ deviceMargin)
$steps \leftarrow floor(SNR_{margin}/3)$
while steps > 0 **and** SF $> SF_{min}$ **do**
    $SF \leftarrow SF - 1$
    $steps \leftarrow steps - 1$
while steps > 0 **and** TP $> TP_{min}$ **do**
    $TP \leftarrow TP - 3$
    $steps \leftarrow steps - 1$
while steps < 0 **and** TP $< TP_{max}$ **do**
    $TP \leftarrow TP + 3$
    $steps \leftarrow steps + 1$

6.5 Předem předpřipravené simulace

V neposlední řadě obsahuje FloRa již předem vytvořené simulace. Jedna z nich je definována v loRaNetworkTest.ini souboru. Skládá se z deseti koncových zařízení, jednoho koncentrátoru a jednoho síťového serveru. Koncové zařízení jsou rozmístěné náhodně v čtvercové oblasti. Každé z nich odešle paket po uplynutí času získaného z exponenciálního rozdělení s průměrem 100 sekund. Dále je každému z nich přidělen z dostupné nabídky nahodný rozprostírací faktor i přenosový výkon. ADR mechanismus mají vypnutý. Ten je vypnutý i v síťovém serveru. Celá simulace trvá 7 dní s tím, že jeden den je určený pro přípravu sítě, tzv. *"warm-up period."* Konfigurace backhaul sítě je definována v souboru cloudDelays.xml. Spojení backhaul sítě obsahuje soubor package.ned [19]



Obrázek 6.1: loRaNetworkTest.ini

6.6 Přidání FloRa framework do OMNeT++

I v případě FloRy je postup pro oba operační systémy stejný. Lze ji stáhnout na `https://flora.aalto.fi/`. Daný soubor extrahujte do složky *"samples"* ve složce omnetpp-5.2.1 a přejmenujte na "flora."Vložení do OMNeT++ IDE je stejné jako v případě INET frameworku, a tedy - *File* -> *Import* -> rozbalte nabídku *General* -> *Existing Projects to the Workspace* -> v *Select root directory* rozklikněte složku *"samples"* a zvolte FloRu. Ta se zobrazí vlevo v simulacích, klikněte na ní pravým a přidejte referenci na INET: *Properties* -> *Project References* -> zaškrněte *inet* -> potvrďte prostřednictvím *Apply and Close*. Sestavte projekt - *Project* -> *Build*, nebo *Ctrl+B* a vše je připravené k práci.

Kapitola 7

Zprovoznění simulací

V téhle sekci se podíváme na dva příklady využívající FloRa framework. U každého příkladu najdete zadání, postup, vypracování, výsledky a závěr. Zdrojové kódy jsou pak k naleznutí v přílohách.

7.1 LoRaWAN síť s vypnutým ADR mechanismem

V první simulaci se budeme zaobírat LoRaWAN síti o malé rozloze bez ADR mechanismu, obsahující 300 koncových zařízení. Zaměříme se hlavně na to, jak moc velká úspěšnost bude pro doručení paketu z koncového zařízení do síťového serveru.

7.1.1 Zadání

Vytvořte LoRaWAN síť v softwaru OMNeT++ využitím FloRa framework. Síť se bude skládat z 300 koncových zařízení, 1 koncentrátoru, 1 síťového serveru a backhaul sítě.

Síť bude velká 480m, proto zvolte konfiguraci *"LoRaLogNormalShadowing." "Warmup-period"* nastavte na 2 dny a dobu pro trvání simulace na 9 dní. Parametr sigma bude 3,57 a minimální časový interval, kdy je možné považovat dva překrývající se signály jako rušivé vůči sobě nastavte na 0.

V síti bude náhodně rozmístěno 300 koncových zařízení, pro které zvolte následující parametry: vypnutý ADR mechanismus, nekonečný počet paketů k odeslání, časy pro odesílání paketů 500 sekund, šířku pásma 125 KHz, náhodný rozprostírací faktor, náhodný vysílací vyýkon dle vzorce 2+3*x, kde x je číslo od 0 do 4, code-rate zvolte 4. Pro simulování spotřeby energie využijte soubor *energyConsumptionParameters.xml*.

Koncentrátor umístěte doprostřed sítě. Pro simulaci backhaul sítě použijte soubor *cloudDelays.xml*. Vypněte ADR mechanismus i pro síťový server. Simulaci opakujte 10x a uveďte výsledky z naměřených dat.

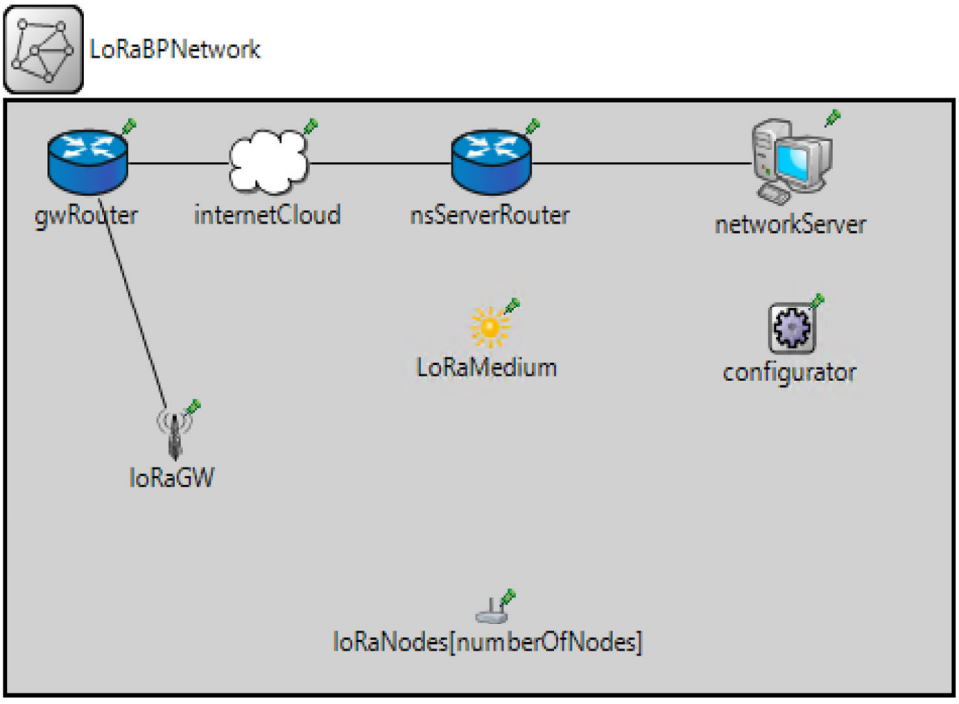
7.1.2 Postup

1. Spusťte program OMNeT++ a v něm vytvořte nový projekt: *File -> New -> OMNeT++ Project* -> zadejte jméno vašeho projektu -> *Next* -> zvolte *Empty project with 'src' and 'simulations' folders* -> *Finish*

2. Ve vámi vytvořeném projektu přidejte referenci na FloRa framework: klikněte na něj pravým tlačítkem myši -> *Properties* -> *Project References* -> zaškrněte *flora* -> potvrďte prostřednictvím *Apply and Close*.

3. Zkopírujte soubory *cloudDelays.xml* a *energyConsumptionParameters.xml* z projektu *flora* a vložte do vámi vytvořeného projektu, do *"simulations,"* nebo je vytvořte a zkopírujte z příloh A a B.

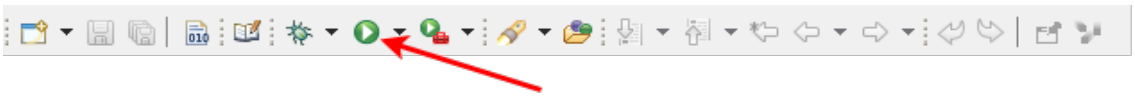
4. Přejděte do souboru *package.ned* a v něm vytvořte síť jako na obrázku 7.1. K jejímu vytvoření použijte moduly *LoRaNode*, *LoRaGW*, *LoRaMedium*, *StandardHost*, *IPv4NetworkConfigurator*, *InternetCloud* a *Router* v nabídce *"Submodels."* K propojení modulů použijte gigabitové ethernetové kabely. Nyní přejděte do zdrojového kódu a změňte *LoRaNode* na pole. Celý zdrojový kód lze nalézt v příloze C.



Obrázek 7.1: Náhled konstruované LoRaWAN sítě

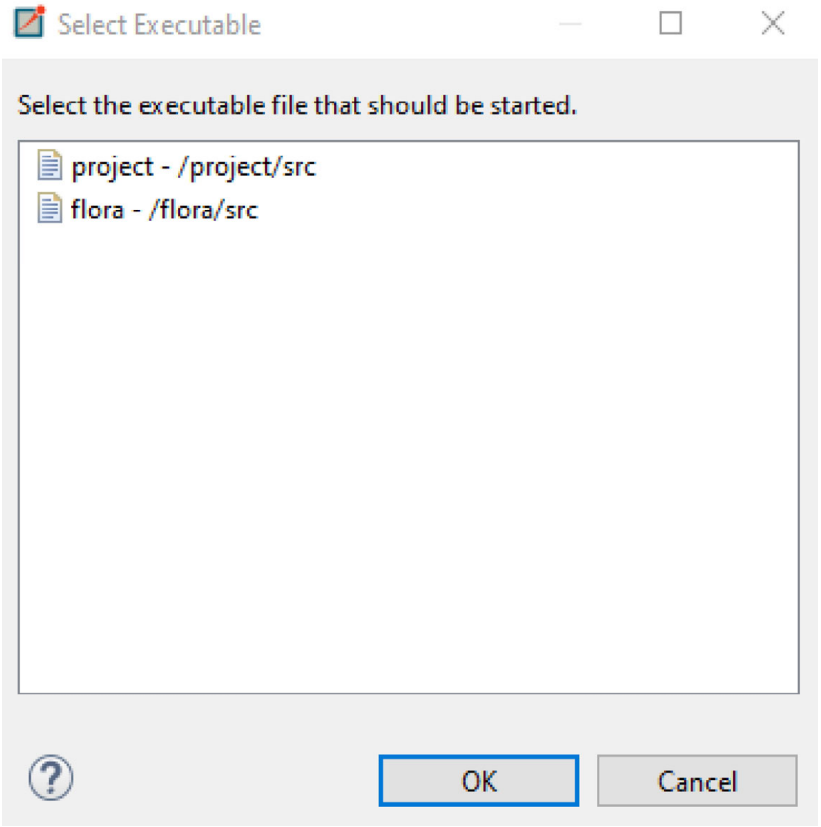
5. Dále přejděte do souboru *omnetpp.ini* a nakonfigurujte v něm parametry sítě uvedené v zadání (příloha D).

6. Jakmile vše dokončíte, spusťte simulaci tlačítkem *"Run"*.



Obrázek 7.2: Spuštění simulace

7. Pokud se vám zobrazí nabídka, jako na obrázku 7.3. Zvolte možnost s florou a dejte *"OK."*



Obrázek 7.3: Nabídka Select Executable

8. Dále budete dotázani, zda chcete přejít do *"release"* módu, zvolte *"No"* a poté se vám otevře simulační prostředí.

9. Zvolte potřebnou konfiguraci a spusťte simulaci. Proces opakujte ještě 9x.

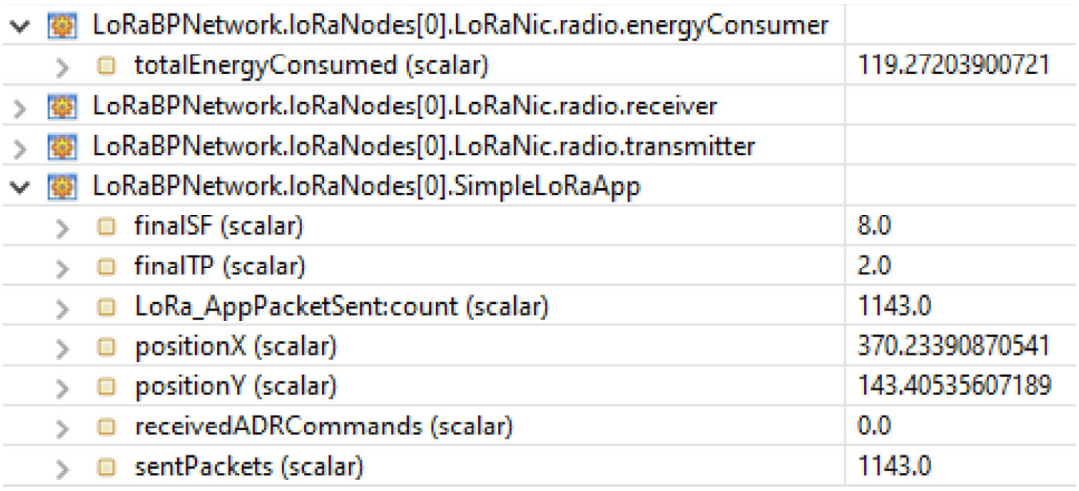
10. Po dokončení desátého cyklu exportujte výsledky do .anf souboru a zanalyzujte výsledky.

7.1.3 Vypracování

V následující sekci se podíváme na vypracování pro první úlohu, kde se zaměříme na naměřené hodnoty koncových zařízení, koncentrátoru a síťového serveru.

7.1.3.1 Koncová zařízení

Jako první se zaměříme na koncová zařízení, konkrétně na moduly *"energyConsumer"* a *"SimpleLoRaApp."*



Obrázek 7.4: Moduly energyConsumer a SimpleLoRaApp

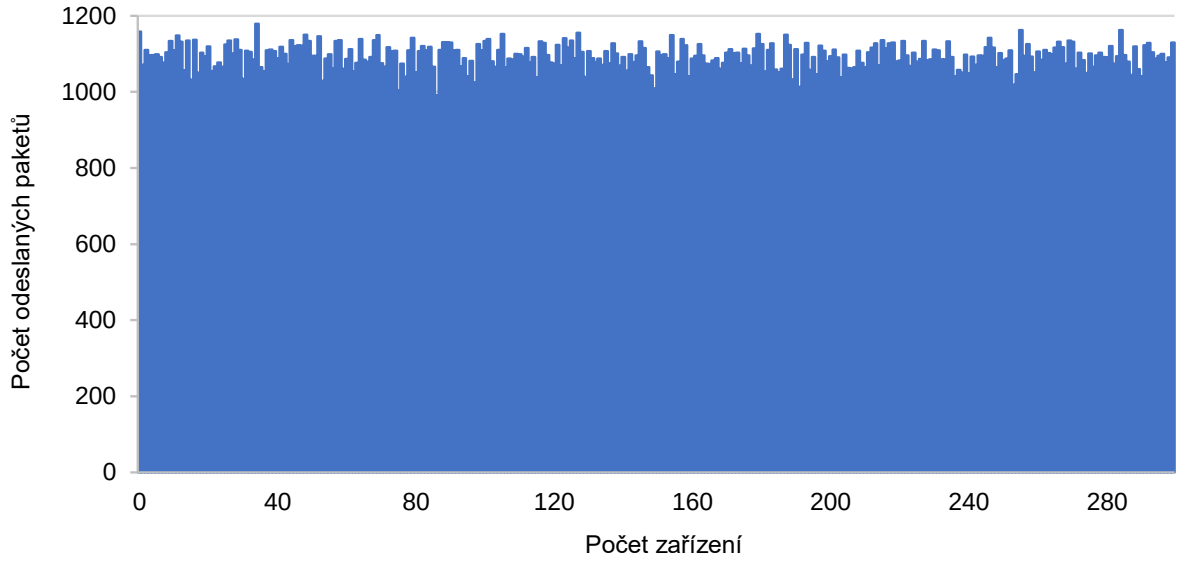
V tabulce 7.1 lze vidět počet zařízení, které využívaly konkrétní rozprostírací faktor (*"finalSF"*) a jejich procentuální zastoupení.

Počet odeslaných paketů každého koncového zařízení (*"LoRa_AppPacketSent"*) je znázorněno v grafu 7.5. Průměrně každé zařízení odeslalo 1095 paketů.

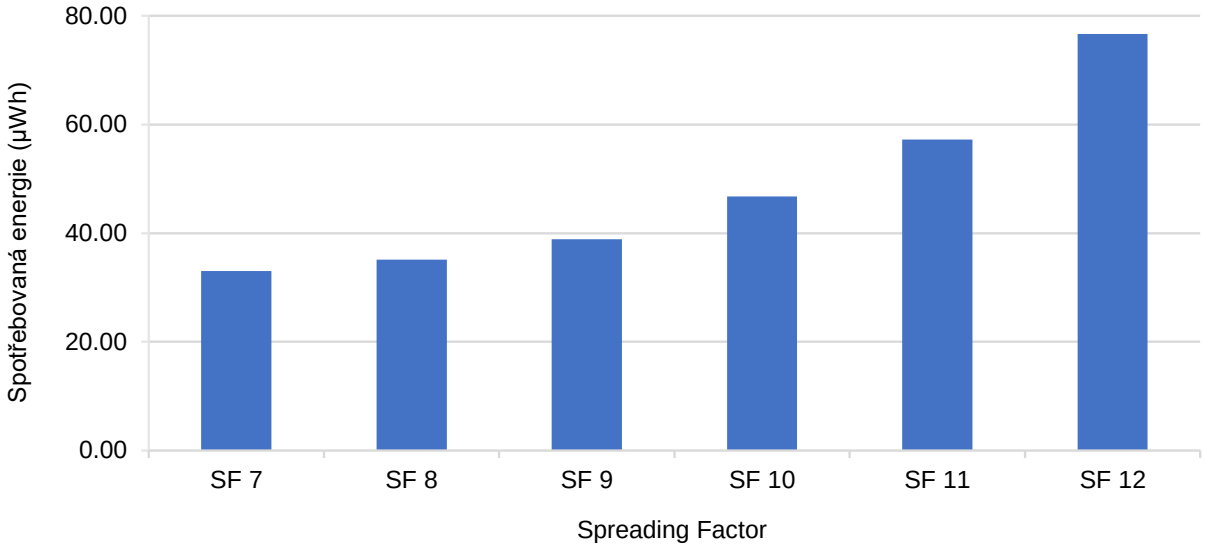
Graf 7.6 zobrazuje spotřebovanou energii v závislosti na rozprostíracím faktoru (*"totalEnergyConsumed,"*) průměrná spotřeba činila 48 μWh.

SF	Počet zařízení	Procentuální rozložení
7	50,5	17%
8	50,3	17%
9	50,8	17%
10	53,1	17%
11	47,6	16%
12	47,7	16%

Tabulka 7.1: LoRaWAN síť s vypnutým ADR mechanismem - rozložení rozprostíracího faktoru



Obrázek 7.5: LoRaWAN síť s vypnutým ADR mechanismem - počet odeslaných paketů na zařízení



Obrázek 7.6: LoRaWAN síť s vypnutým ADR mechanismem - spotřeba energie pro jednotlivé rozprostírací faktory

7.1.3.2 Koncentrátor a síťový server

Pro výsledky koncentrátoru využijeme modul *"radio."*

Koncentrátor celkově přijal 123370 paketů (*"LoRaGWRadioReceptionFinishedCorrect"*) z 328510

LoRaBPNetwork.loRaGW[0].LoRaGWNic.radio	
bitErrorRate:histogram (histogram)	NaN (0) [30 bins]
DER - Data Extraction Rate (scalar)	0.35722631414429
LoRaGWRadioReceptionFinishedCorrect:count (scalar)	117710.0
LoRaGWRadioReceptionStarted:count (scalar)	329511.0
minSNIR:histogram (histogram)	2.4455177431108...
packetErrorRate:histogram (histogram)	NaN (0) [30 bins]
radioMode:count (scalar)	0.0
receptionState:count (scalar)	0.0
symbolErrorRate:histogram (histogram)	NaN (0) [30 bins]
transmissionState:count (scalar)	0.0

Obrázek 7.7: Modul radio

odeslaných koncovými zařízeními (*"LoRaGWRadioReceptionStarted,"*) což činí poměr přijatých paketů vůči všem odeslaným 37,6% (*"DER"*).

V neposlední řadě se podíváme na modul *"udpApp."* v síťovém serveru.

LoRaBPNetwork.networkServer.udpApp[0]	
counterUniqueReceivedPacketsPerSF SF7 (scalar)	3782.0
counterUniqueReceivedPacketsPerSF SF8 (scalar)	10680.0
counterUniqueReceivedPacketsPerSF SF9 (scalar)	21963.0
counterUniqueReceivedPacketsPerSF SF10 (scalar)	27131.0
counterUniqueReceivedPacketsPerSF SF11 (scalar)	26791.0
counterUniqueReceivedPacketsPerSF SF12 (scalar)	27363.0
DER SF7 (scalar)	0.064434790016185
DER SF8 (scalar)	0.16185987299759
DER SF9 (scalar)	0.38186560027819
DER SF10 (scalar)	0.48501045782013
DER SF11 (scalar)	0.53677546031937
DER SF12 (scalar)	0.65985820391627
LoRa_NS_DER (scalar)	0.35722631414429
LoRa_ServerPacketReceived:count (scalar)	117710.0
receivedRSSI (histogram)	-126.87553384874...
Send ADR for node (scalar)	0.0
Send ADR for node (scalar)	0.0

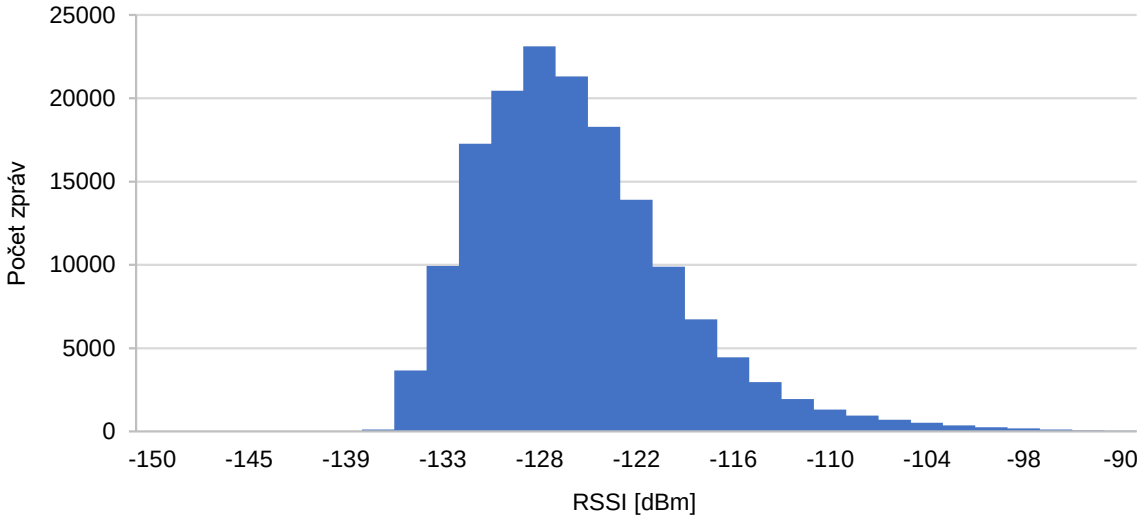
Obrázek 7.8: Modul udpApp

DER síťového serveru je stejně jako u koncentrátoru - 37,6%. Hodnoty DER a počet přijatých paketů pro každý rozprostírací faktor lze vidět v tabulce 7.2.

Histogram 7.9 znázorňuje počet zpráv a jim odpovídající RSSI. Průměrná hodnota RSSI činila -126 dBm, nejmenší -137 dBm a nejvyšší -84 dBm.

SF	DER [%]	Počet přijatých paketů
7	10	6195
8	20	11864
9	35	20361
10	49	28609
11	59	29355
12	63	26985

Tabulka 7.2: LoRaWAN síť s vypnutým ADR mechanismem - DER a počet přijatých paketů pro každý rozprostírací faktor



Obrázek 7.9: LoRaWAN síť s vypnutým ADR mechanismem - RSSI histogram

7.1.4 Závěr

Koncová zařízení průměrně zaslala 1095 paketů, maximum tvořilo číslo 1303 a minimum 831. Rozprostírací faktory byly víceméně rovnoměrně rozložené díky funkci ze zdrojového kódu. Podíváme-li se na spotřebu energie, lze konstatovat, že čím větší rozprostírací faktor, tím větší i spotřeba. Zařízení se SF 12 a SF 11 předčily průměrnou hodnotu spotřeby (48 μWh), zatímco zařízení se SF7,8,9 byly výrazně nižší, zařízení se SF 10 byly těsně pod průměrem.

Poměr přijatých paketů vůči všem odeslaným paketům koncovými zařízeními činil 37,6%, to znamená, že víceméně dvě třetiny paketů nebyly doručeny. Důvodem jsou kolize, náhodný rozprostírací faktor a náhodný přenosový výkon. To nás přináší k tabulce 7.2, kde lze vidět, že nejhorší hodnoty DER zaznamenaly zařízení se SF 7 a SF 8. Naopak zařízení se SF 12 zaznamenaly nejlepší DER s 63%.

7.2 LoRaWAN síť se zapnutým ADR mechanismem

V druhé úloze využijeme stejnou konfiguraci jako v první. S rozdílem, že tentokrát zapneme ADR mechanismus v koncových zařízeních i v síťovém serveru a porovnáme naměřené hodnoty získané z obou simulací.

7.2.1 Zadání

Využijte vámi vytvořenou síť z první úlohy, ale zapněte ADR mechanismus v koncovém zařízení a ADR+ mechanismus v síťovém serveru.

7.2.2 Postup

1. Přejděte do souboru *omnetpp.ini* a přidejte do něj řádek ***.networkServer.udpApp[0].adrMethod = ${"avg"}*.

2. Dále zapněte ADR mechanismus pro koncová zařízení i síťový server, viz. příloha D.

3. Simulaci opakujte 10x, uveďte výsledky z naměřených dat a porovnejte je s výsledky z první úlohy.

7.2.3 Vypracování

Vypracování pro druhou úlohou je víceméně totožné jako v předchozím případě. Pro výsledné hodnoty využijeme stejné moduly, akorát přidáme ještě údaje o přijatých ADR paketech, které lze najít v modulu *"SimpleLoRaApp"* v koncovém zařízení.

7.2.3.1 Koncová zařízení

Tabulka 7.3 obsahuje konkrétní počet zařízení pro daný rozprostírací faktor s následným procentuálním zastoupením.

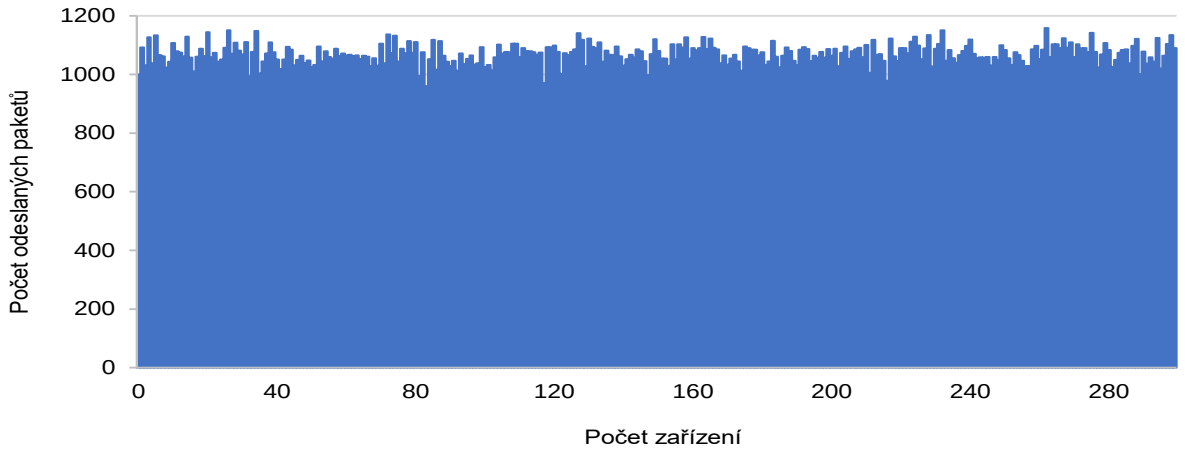
Graf 7.10 znárorňuje počet odeslaných paketů všech koncových zařízení. Zde platilo, že každé zařízení odeslalo průměrně 1071 paketů.

V grafu 7.11 lze vidět spotřebu energie pro jednotlivé rozprostírací faktory. Průměrná spotřeba pro LoRaWAN síť se zapnutým ADR mechanismem činila 52 μWh.

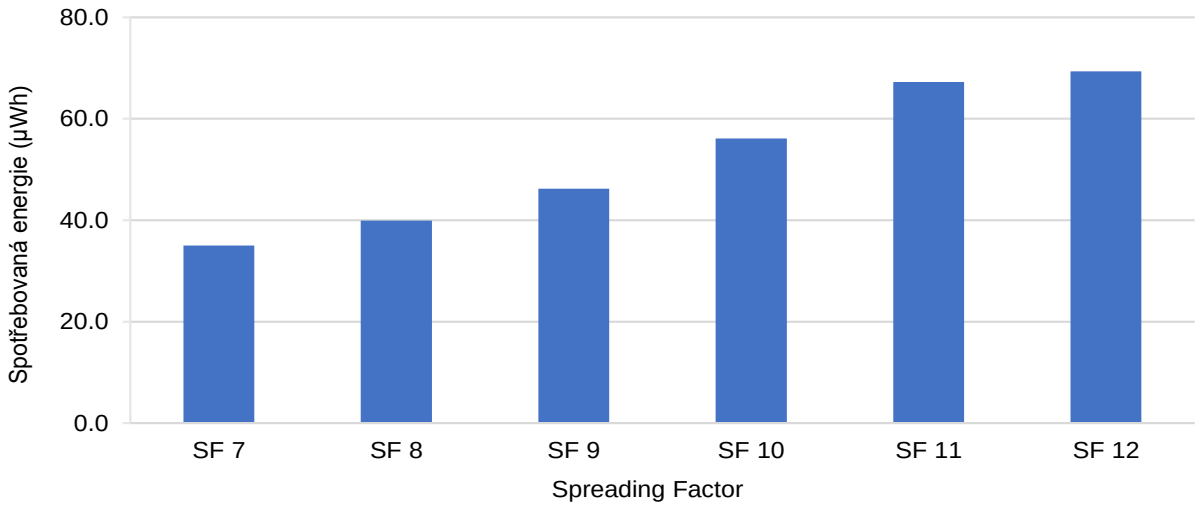
Na jedno koncové zařízení spadá průměrně 29 přijatých ADR paketů ze síťového serveru. V grafu 7.12 jde vidět každé zařízení a počet paketů, které přijal.

SF	Počet zařízení	Procentuální rozložení
7	39	13%
8	31,5	10%
9	49,5	16%
10	73,8	25%
11	55,9	19%
12	50,3	17%

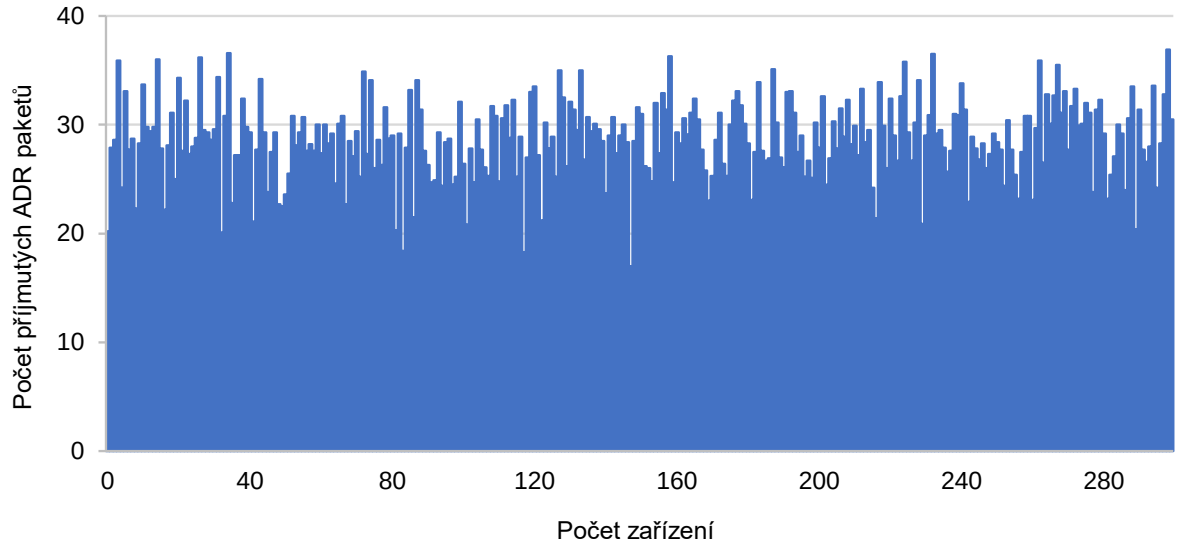
Tabulka 7.3: LoRaWAN síť se zapnutým ADR mechanismem - rozložení rozprostíracího faktoru



Obrázek 7.10: LoRaWAN síť se zapnutým ADR mechanismem - počet odeslaných paketů na zařízení



Obrázek 7.11: LoRaWAN síť se zapnutým ADR mechanismem - spotřeba energie pro jednotlivé rozprostírací faktory



Obrázek 7.12: LoRaWAN síť se zapnutým ADR mechanismem - počet přijatých ADR paketů na zařízení

7.2.3.2 Koncentrátor a síťový server

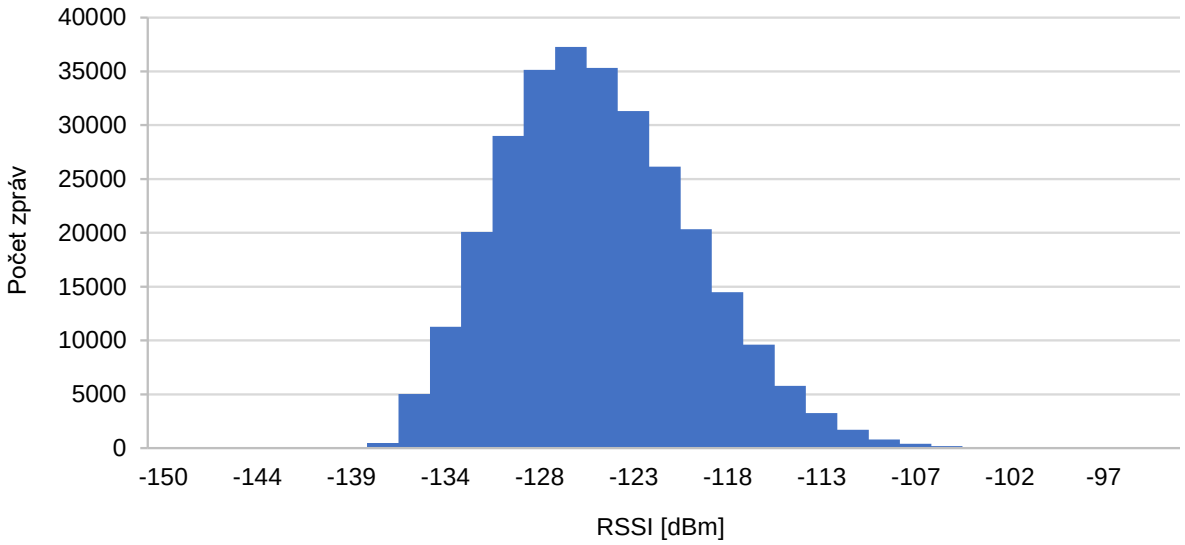
DER koncentrátoru činilo 75%. Z celkově 321233 odeslaných paketů z koncových zařízení jich přijal 240161.

Síťový server, stejně jako koncentrátor, přijal 75% paketů. Hodnota DER a počet přijatých paketů pro každý rozprostírací faktor je znázorněn v tabulce 7.4

SF	DER [%]	Počet přijatých paketů
7	82	31451
8	83	29157
9	82	46107
10	79	61845
11	76	40316
12	52	31285

Tabulka 7.4: LoRaWAN síť se zapnutým ADR mechanismem - DER a počet přijatých paketů pro každý rozprostírací faktor

Na histogramu 7.13 lze vidět počet zpráv a jim odpovídající RSSI. Průměrná hodnota RSSI činila -126 dBm, nejmenší -137 dBm a nejvyšší -88 dBm.



Obrázek 7.13: LoRaWAN síť se zapnutým ADR mechanismem - RSSI histogram

7.2.4 Závěr

V druhé úloze je počet odeslaných paketů koncovými zařízeními podobný jako v první simulaci. První změna nastává u rozprostíracího faktoru, kde zapracoval ADR+ mechanismus. Zatímco v první úloze byly hodnoty SF víceméně rovnoměrne rozložené, zde viditelně převládá SF 10 s 25%. Na opačné straně využila zařízení SF 8 jen v 10% případech. Co se porovnání spotřeby týká, tak síť se zapnutým ADR+ mechanismem vykazovala cca o 4 μWh vyšší spotřebu. Každé zařízení průměrně přijalo 29 ADR příkazů a RSSI bylo u obou úloh podobné.

Síť jako celek fungovala daleko lépe se zapnutým ADR+ mechanismem než bez něho, kdy bylo přijato 75% všech odeslaných paketů z koncových zařízení. Dalším rozdílem oproti první úloze je hodnota DER pro jednotlivé rozprostírací faktory. V předchozím případě vykazovaly vyšší procenta přijatých paketů zařízení s vyšším rozprostíracím faktorem, zatímco zde je trend opačný. Celkově pro všechny rozprostírací faktory byla úspěšnost přijatých paketů vysoká, s výjimkou SF 12, kde parametr DER dosáhl 52%. Důvodem je, že SF 12 má vysoké ToA, a pokud je síť správně nastavena (jako v tomhle případě), je daleko větší šance, že právě paket odeslaný z koncového zařízení se SF 12 zkoliduje s nějakým jiným. Důvodem proč byl v první simulaci trend opačný je ten, že zařízení s vyšším rozprostíracím faktorem nám poskytovalo dosah, který u několika zařízení chyběl - například mohla nastat situace, že se ocitlo zařízení s nízkým rozprostíracím faktorem a nízkým přenosovým výkonem na krají sítě, a tudíž nikdy nedosáhlo na koncentrátor.

Co se porovnání obou úloh týká, je zřejmé, že síť se zapnutým ADR+ mechanismem vykazovala daleko lepší výsledky. Sice vzrostla o trochu spotřeba energie, ale za to přijala skoro o 40% víc paketů.

Kapitola 8

Závěr

Cílem práce bylo seznámit čtenáře se simulační knihovnou FloRa a zhotovit dvě laboratorní úlohy do odborného předmětu. Práce čtenáře postupně informuje o základních principech technologie LoRaWAN, prostředí OMNeT++ a FloRy, na základě kterých by měl být schopný vytvořit svou vlastní simulaci a sám zhodnotit naměřené výsledky.

Obě vytvořené úlohy jsou pojaty spíš náučnou formou, kdy je čtenář krok za krokem proveden k vytvoření simulace a následnému zpracování naměřených dat.

První úloha se zaobírá LoRaWAN sítí o rozloze 480 metrů s 300 zařízeními a slouží spíš jako praktický úvod do prostředí OMNeT++ a FloRa frameworku. Druhá simulace postupuje v podobné myšlence, ale rozšiřuje předešlou úlohu o ADR+ mechanismu. Zde bylo ukázano, jaký dopad měl na celou síť a bylo prokázano, že síť s ADR+ mechanismem jeví daleko lepší výsledky.

V rámci FloRy jsem také provedl zátěžový test, ke kterému jsem využil zdrojový kód z druhé úlohy. V zdrojovém kódu jsem změnil počet koncových zařízení z 300 na 5000 a každému přiřadil 15 paketů k odeslání. Tahle vytvořená simulace proběhla v pořádku a zvládla zaznamenat výsledky.

V neposlední řadě jsem si všiml, že v době dokončování bakalářské práce, vydali vývojáři novou verzi FloRy. Verze je označená jako 1.0.0, přináší nové funkce a lze ji využít ve verzi 6 pro OMNeT++ a ve verzi 4.3.1 pro INET. Verzi 0.8 pro FloRu, kterou jsem využíval, je možné najít na GitHubu vývojářů. Odkaz na GitHub lze nalézt na jejich webových stránkách, viz [19]. Samotná instalace softwaru OMNeT++ a přidání obou frameworků zůstává stejná, jen s jinými verzemi.

S příchodem nové verze přichází i myšlenka, jestli se jedná o jednu z několika dalších. V tom případě by se mohla tahle práce i vytvořené úlohy časem stát neaktuální a potřebovaly by dostat nutných změn.

Literatura

1. MEKKI, Kais; BAJIC, Eddy; CHAXEL, Frederic; MEYER, Fernand. *A comparative study of LPWAN technologies for large-scale IoT deployment* [online] [cit. 2020-10-22]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2405959517302953.
2. RAZA, Usman; KULKARNI, Parag; SOORIYABANDARA, Mahesh. *Low Power Wide Area Networks: An Overview* [online] [cit. 2020-10-22]. Dostupné z: https://ieeexplore.ieee.org/document/7815384.
3. LORA ALLIANCE. *A technical overview of LoRa® and LoRaWAN™* [online] [cit. 2020-11-04]. Dostupné z: https://lora-alliance.org/resource-hub/what-lorawanr.
4. SEMTECH DEVELOPER PORTAL. *LoRa and LoRaWAN: Technical overview* [online] [cit. 2020-10-25]. Dostupné z: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/.
5. MROUE, H.; NASSER, A.; PARREIN, B.; HAMRIOUI, S.; MONA-CRUZ, E.; ROUYER, G. *Analytical and Simulation study for LoRa Modulation* [online] [cit. 2020-10-25]. Dostupné z: https://ieeexplore.ieee.org/document/8464879.
6. LORA ALLIANCE. *RP002-1.0.1 LoRaWAN Regional Parameters* [online] [cit. 2020-11-14]. Dostupné z: https://lora-alliance.org/resource-hub/rp2-101-lorawanr-regional-parameters-0.
7. SEMTECH DEVELOPER PORTAL. *Understanding ADR* [online] [cit. 2020-11-08]. Dostupné z: https://lora-developers.semtech.com/library/tech-papers-and-guides/understanding-adr.
8. LORA ALLIANCE. *About LoRa Alliance®* [online] [cit. 2020-11-04]. Dostupné z: https://lora-alliance.org/about-lora-alliance.
9. LORA ALLIANCE. *About LoRaWAN®* [online] [cit. 2020-11-05]. Dostupné z: https://lora-alliance.org/about-lorawan.
10. THE THINGS NETWORK. *Duty Cycle* [online] [cit. 2020-12-13]. Dostupné z: https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html.

11. ČESKÝ TELEKOMUNIKAČNÍ ÚŘAD. *Využívání vymezených rádiových kmitočtů* [online] [cit. 2021-04-26]. Dostupné z: https://www.ctu.cz/vyuzivani-vymezenych-radiovych-kmitoctu.

12. LORA ALLIANCE. *LoRaWAN™ 1.0.3 Specification* [online] [cit. 2020-11-16]. Dostupné z: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf.

13. TUTORIALWING. *Pure Aloha Protocol Tutorial* [online] [cit. 2020-12-11]. Dostupné z: https://tutorialwing.com/pure-aloha-protocol-tutorial-with-example/.

14. SEMTECH DEVELOPER PORTAL. *An In-depth look at LoRaWAN® Class A Devices* [online] [cit. 2020-12-11]. Dostupné z: https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-a-devices/.

15. ČESKÝ RADIOKOMUNIKAČNÍ ÚŘAD. *Technické aspekty technologie LoRa* [online] [cit. 2020-11-16]. Dostupné z: https://pripoj.me/technicke-aspekty-technologie-lora/.

16. OMNET++. *Simulation Manual* [online] [cit. 2021-03-27]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/.

17. INET FRAMEWORK. *What Is INET Framework?* [Online] [cit. 2021-04-07]. Dostupné z: https://inet.omnetpp.org/Introduction.html.

18. OMNET++. *Installation Guide* [online] [cit. 2021-02-01]. Dostupné z: https://doc.omnetpp.org/omnetpp/InstallGuide.pdf.

19. SLABICKI, Mariusz; PREMSANKAR, Gopika. *FLoRa - A Framework for LoRa simulations* [online] [cit. 2021-02-04]. Dostupné z: https://flora.aalto.fi/.

20. BOR, Martin C.; ROEDIG, Utz; VOIGT, Thiemo; ALONSO, Juan M. *Do LoRa Low-Power Wide-Area Networks Scale?* [Online] [cit. 2021-04-25]. Dostupné z: https://doi.org/10.1145/2988287.2989163.

21. SLABICKI, Mariusz; PREMSANKAR, Gopika; FRANCESCO, Mario Di. *Adaptive Configuration of LoRa Networks for Dense IoT Deployments* [online] [cit. 2021-02-04]. Dostupné z: https://flora.aalto.fi/.

Příloha A

Konfigurace backhaul sítě

```
<internetCloud symmetric="true">
  <parameters name="good">
    <traffic src="**" dest="**" delay="10ms" datarate="1Gbps" drop="uniform(0,1)
        \&lt; 0" />
  </parameters>
</internetCloud>
```

Příloha B

Konfigurace spotřeby energie

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <supplyVoltage value="3.3"/>
   <receiverReceivingSupplyCurrent value="9.7"/>
   <receiverBusySupplyCurrent value="9.7"/>
   <idleSupplyCurrent value="0.0001"/>
   <txSupplyCurrents>
      <txSupplyCurrent txPower="2" supplyCurrent="24"/>
      <txSupplyCurrent txPower="3" supplyCurrent="24"/>
      <txSupplyCurrent txPower="4" supplyCurrent="24"/>
      <txSupplyCurrent txPower="5" supplyCurrent="25"/>
      <txSupplyCurrent txPower="6" supplyCurrent="25"/>
      <txSupplyCurrent txPower="7" supplyCurrent="25"/>
      <txSupplyCurrent txPower="8" supplyCurrent="25"/>
      <txSupplyCurrent txPower="9" supplyCurrent="26"/>
      <txSupplyCurrent txPower="10" supplyCurrent="31"/>
      <txSupplyCurrent txPower="11" supplyCurrent="32"/>
      <txSupplyCurrent txPower="12" supplyCurrent="34"/>
      <txSupplyCurrent txPower="13" supplyCurrent="35"/>
      <txSupplyCurrent txPower="14" supplyCurrent="44"/>
   </txSupplyCurrents>
</root>
```

Příloha C

Konfigurace v souboru package.ned

```
import inet.applications.udpapp.UDPBasicApp;
import inet.node.inet.Router;
import inet.node.internetcloud.InternetCloud;
import loranetwork.LoRaPhy.LoRaMedium;
import loranetwork.LoraNode.LoRaNode;
import loranetwork.LoraNode.LoRaGW;
import inet.node.inet.StandardHost;
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.node.ethernet.Eth1G;

network LoRaBPNetwork
{
     parameters:
        int numberOfNodes = default(1);
        int numberOfGateways = default(1);
        int networkSizeX = default(480);
        int networkSizeY = default(480);
    @display("bgb=468,309");
    submodules:
        loRaNodes[numberOfNodes]: LoRaNode {
            @display("p=240,230");
        }
        loRaGW: LoRaGW {
            @display("p=83.664,150.192;is=s");
        }
        LoRaMedium: LoRaMedium {
```

```
            @display("p=239,102");
        }
        networkServer: StandardHost {
            parameters:
                @display("p=389,27");
        }
        configurator: IPv4NetworkConfigurator {
            parameters:
                assignDisjunctSubnetAddresses = false;
                @display("p=389,102");
        }
        internetCloud: InternetCloud {
            @display("p=130,27");
        }
        gwRouter: Router {
            @display("p=40,27");
        }
        nsServerRouter: Router {
            @display("p=240,27");
        }
    connections:
        networkServer.ethg++ <--> Eth1G <--> nsServerRouter.ethg++;
        nsServerRouter.pppg++ <--> Eth1G <--> internetCloud.pppg++;
        internetCloud.pppg++ <--> Eth1G <--> gwRouter.pppg++;
        gwRouter.ethg++ <--> Eth1G <--> loRaGW.ethg++;

}
```

Příloha D

Konfigurace v souboru omnetpp.ini

```
[General]
#uvedeni site z package.ned
network = LoRaBPNetwork
#vypnuti snimani vektorovych souboru
**.vector-recording = false

#zakladni nastaveni pro LoRa koncentrator
**.loRaGW.numUdpApps = 1
**.loRaGW.packetForwarder.localPort = 2000
**.loRaGW.packetForwarder.destPort = 1000
**.loRaGW.packetForwarder.destAddresses = "networkServer"

#zakladni nastaveni pro LoRa sitovy server
**.networkServer.numUdpApps = 1
**.networkServer.udpApp[0].typename = "NetworkServerApp"
**.networkServer.udpApp[0].destAddresses = "loRaGW"
**.networkServer.udpApp[0].destPort = 2000
**.networkServer.udpApp[0].localPort = 1000

#argument dava na vyber pri sputeni simulace mezi ADR a ADR+
**.networkServer.udpApp[0].adrMethod = ${adrMethod="avg","max"}

#pocet koncovych zarizeni
**.numberOfNodes = 300
#pocet paketu k zaslani (0 znaci nekonecno)
**.numberOfPacketsToSend = 0
```

```
#doba po kterou se nezaznamenavaji vysledky (ve dnech)
warmup-period = 2d
#doba po kterou simulace bezi (ve dnech)
sim-time-limit = 9d
#rozliseni simulace
simtime-resolution = -11

#definice jak casto se pakety budou posilat
**.timeToFirstPacket = exponential(500s)
**.timeToNextPacket = exponential(500s)

#nastaveni pro koncove zarizeni
#pokud false, koncova zarizeni jsou zobrazena na svych skutecnych mistech, pokud
    true, koncova zarizeni jsou zobrazena na jednom miste jako pole koncovych
    zarizeni (pro 2 a vice koncovych zarizeni)
**.loRaNodes[*].**.initFromDisplayString = false
#kazdemu zarizeni je pridan nahodne jeden spreading factor z hodnot 7 až 12
**.loRaNodes[*].**initialLoRaSF = intuniform(7, 12)
#bandwidth
**.loRaNodes[*].**initialLoRaBW = 125 kHz
#code rate
**.loRaNodes[*].**initialLoRaCR = 4
#obdobne jako u rozprostiraciho faktoru, vysilaci vykon je minimalne 2 dBm a
    maximalne 14 dBm
**.loRaNodes[*].**initialLoRaTP = (2dBm + 3dBm*intuniform(0, 4))

#nastaveni pro spotřebu, jednotlive hodnoty se berou ze souboru
    energyConsumptionParameters.xml, ktere jsou zavisle na vysilacim vykonu, a s
    kterymi se dale pracuje v ramci tridy LoRaEnergyConsumer
**.loRaNodes[*].LoRaNic.radio.energyConsumerType = "LoRaEnergyConsumer"
**.loRaNodes[*].**.energySourceModule = "IdealEpEnergyStorage"
**.loRaNodes[*].LoRaNic.radio.energyConsumer.configFile = xmldoc("
    energyConsumptionParameters.xml")

#konfigurace backhaul site
**.delayer.config = xmldoc("cloudDelays.xml")
#urceni media pro prenos
**.radio.radioMediumModule = "LoRaMedium"
```

```
#minimalni casovy interval, kdy je mozne povazovat dva prekryvajici se signály
    jako rusive vuci sobe
**.minInterferenceTime = 0s

#nastaveni minimalnich hodnot pro velikost oblasti, ve ktere se simulovana sit
    nachazi
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxZ = 0m

#pokud false, koncentratory jsou zobrazeny na svych skutecnych mistech, pokud true
    , koncentratory jsou zobrazeny na jednom miste jako pole koncentratoru (pro 2
    a vice koncentratoru)
**.loRaGW.**.initFromDisplayString = false

#simulace vyuzita v bakalarske praci jako sit s vypnutym ADR mechanismem
[Config FirstSim-NoADR]
#pocet opakovani
repeat = 10
#vypnuti ADR v sitovem serveru
**.networkServer.**.evaluateADRinServer = false
#vypnuti ADR v koncovych zarizenich
**.loRaNodes[*].**.evaluateADRinNode = false
#slouzi k simulacim urcenych v mestske oblasti
**.LoRaMedium.pathLossType = "LoRaLogNormalShadowing"
#pro primestske oblasti je "LoRaPathLossOulu"

#podminky pro prenos - 3,57 jsou pro mestske oblasti v ramci FloRy
    nejnepriznivejsi podminky
**.sigma = 3.57
#v pripade "LoRaPathLossOulu" je pro nejnepriznivejsi podminky hodnota 7,8

#definovani maximalni velikosti oblasti, ve ktere se simulovana sit nachazi
**.constraintAreaMaxX = ${networkSize=480m}
**.constraintAreaMaxY = ${networkSize}
#umisteni koncentratoru doprostred
**.loRaGW.**.initialX = ${networkSize}/2
```

```
**.loRaGW.**.initialY = ${networkSize}/2
#nahodne rozlozeni koncovych zarizeni v ramci definovane oblasti site
**.loRaNodes[*].**.initialX = uniform(0m, ${networkSize})
**.loRaNodes[*].**.initialY = uniform(0m, ${networkSize})

#simulace nevyuzita v bakalarske praci
#sit se zapnutym ADR mechanismem v koncovem zarizeni
[Config SecondSim-ADRNode]
repeat = 10
**.networkServer.**.evaluateADRinServer = false
**.loRaNodes[*].**.evaluateADRinNode = true
**.LoRaMedium.pathLossType = "LoRaLogNormalShadowing"
**.sigma = 3.57
**.constraintAreaMaxX = ${networkSize=480m}
**.constraintAreaMaxY = ${networkSize}
**.loRaGW.**.initialX = ${networkSize}/2
**.loRaGW.**.initialY = ${networkSize}/2
**.loRaNodes[*].**.initialX = uniform(0m, ${networkSize})
**.loRaNodes[*].**.initialY = uniform(0m, ${networkSize})

#simulace nevyuzita v bakalarske praci
#sit se zapnutym ADR mechanismem v koncovem zarizeni i sitovem serveru
[Config ThirdSim-MaxADR]
repeat = 10
**.networkServer.**.evaluateADRinServer = true
**.loRaNodes[*].**.evaluateADRinNode = true
**.LoRaMedium.pathLossType = "LoRaLogNormalShadowing"
**.sigma = 3.57
**.constraintAreaMaxX = ${networkSize=480m}
**.constraintAreaMaxY = ${networkSize}
**.loRaGW.**.initialX = ${networkSize}/2
**.loRaGW.**.initialY = ${networkSize}/2
**.loRaNodes[*].**.initialX = uniform(0m, ${networkSize})
**.loRaNodes[*].**.initialY = uniform(0m, ${networkSize})

#simulace vyuzita v bakalarske praci jako sit se zapnutym ADR mechanismem
#zdrojovy kod se lisi od prvni simulace jen v radcich pro zapnuti ADR mechanismu
#od treti zde uvedene se lisi, ze vyuzivala ADR+ mechanismus
```

```
[Config FourthSim-AvgADR]
repeat = 10
#zapnuti ADR v sitovem serveru
**.networkServer.**.evaluateADRinServer = true
#zapnuti ADR v koncovych zarizenich
**.loRaNodes[*].**.evaluateADRinNode = true
**.LoRaMedium.pathLossType = "LoRaLogNormalShadowing"
**.sigma = 3.57
**.constraintAreaMaxX = ${networkSize=480m}
**.constraintAreaMaxY = ${networkSize}
**.loRaGW.**.initialX = ${networkSize}/2
**.loRaGW.**.initialY = ${networkSize}/2
**.loRaNodes[*].**.initialX = uniform(0m, ${networkSize})
**.loRaNodes[*].**.initialY = uniform(0m, ${networkSize})
```